HORN'S CURVE ESTIMATION THROUGH MULTI-DIMENSIONAL INTERPOLATION

THESIS

Andrew L. Bigley, Captain, USAF

AFIT-ENS-13-M-01

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENS-13-M-01

HORN'S CURVE ESTIMATION THROUGH MULTI-DIMENSIONAL INTERPOLATION

THESIS

Department of Operational Sciences

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Operational Research

Andrew L. Bigley, BS

Captain, USAF

March 2013

AFIT-ENS-13-M-01

HORN'S CURVE ESTIMATION THROUGH MULTI-DIMENSIONAL INTERPOLATION

Andrew L. Bigley, BS
Captain, USAF

Approved:

_____ _____
Dr. Kenneth W. Bauer (Member) date

_____ _____
Dr. Mark A. Friend, Lt Col, USAF (Member) date

Abstract

A well-known multivariate data reduction method is principal components analysis (PCA). PCA transforms the variables under study into a set of components that are used to summarize the variation among the variables. The benefit is the dimension of the data may be reduced by the descriptive power of the components, permitting tractable analysis on large and messy datasets. Integral to successful PCA is determining when to stop extracting components – the matter is not a trivial one.

A method that consistently produces reliable component extraction estimates is Horn's test, named after researcher John L. Horn who introduced the technique in 1965. The result is Horn's curve, a graphical indicator used to make a dimensionality assessment for any n x p matrix. The drawback of Horn's test is it requires – for each size n x p study – a large amount of random data to evaluate the hidden component structure.

Leveraging the flexibility and power of the MATLAB software package, a lookup table interpolates nearest neighbor searches of pre-processed mean eigenvalue data to provide real-time results for datasets up to 1,000 variables on 7,000 samples. The methodology is extended to a linear regression second-order model producing Horn's curve, significantly reducing the required size of the lookup table with no loss of resolution into the dimensionality estimate.

Dedication

To my mother, who taught me to be responsible member of society and was constantly putting the best-interests of her children in front of everything else, even if it was uncomfortable in one way or another;

To my father, who showed me how to be a good dad long before I became one and is always willing to listen, without judging or giving advice;

To my daughter, who reminds me that every hour of a day is not all about work and to stop and have fun now and then, and is growing up way too fast;

To my significant other for her patience and care in letting me do what I needed to do to be successful at AFIT;

and to all the family and friends that I had to miss seeing as often as we would have liked over the last eighteen months–thank you for the encouragement that we will visit together soon.

Acknowledgments

I would like to express my sincere thanks to my research advisor, Dr. Kenneth Bauer, for his wisdom and guidance throughout the course of this thesis effort. It was his spark and motivation that led me this direction. His talent as an academician extends well beyond the storehouse of knowledge he possesses.

I am also grateful to my reader and student advisor, Lt Col (Dr) Mark Friend, who helped me uncover the roadmap to being a successful student at AFIT; the mentoring sessions were invaluable! His passion for teaching and instilling learning in others cannot be overstated.

Finally, a big thank you goes to Trevor Bihl at the Sensor Fusion Lab. His grasp of the inner workings of MATLAB and the ability to suggest new coding avenues when things bogged down were pivotal to achieving success with the algorithms.

The comments and questions they asked of me provided insight into dark corners of the problem statement and research objective I had not considered, and for them, this work has certainly been made stronger. Any shortcomings, omissions, or errors are solely my responsibility.

Andrew L. Bigley

Table of Contents

Page

List of Figures

Page

List of Tables

Page

HORN'S CURVE ESTIMATION THROUGH MULTI-DIMENSIONAL INTERPOLATION

I. Introduction

1.1. Background

Prevalent in the fields of applied science is the need to conduct experiments, collect data, and draw meaningful conclusions from the observations. Quite often data are multivariate and the simultaneous interactions among all the variables are of interest. Datasets may consist of hundreds of variables (p) and tens of thousands of samples (n). Contemporary data storage capacity and computer processing power means it is possible to access trillions of data bits with little effort or significant cost. Field work is still alive and well – designing experiments, conducting tests, and recording results are still part of a scientist's job description – and the ability to collaborate and instantaneously share information has transformed almost all walks of research. The researcher would probably find it useful to describe the relationship between variables without having to report each and every raw combination of the data. Ideally, no data would be discarded yet a way to summarize the important information is needed.

1.2. Principal Components Analysis

One of the most commonly used data reduction techniques is called principal components analysis (PCA). The word 'principal' is used to mean that some components are significant and should be used for further analysis (a process called extraction). Components not significant should be discarded and excluded from additional analysis.

When the principal components have been identified, we speak of the dimensionality of the data. For example, extracting four principal components of twelve total components results in a dimensionality of four.

The goal of PCA is to describe as much of the total variance among the variables as possible by using a smaller number of linear combinations – the principal components – of the variables without losing useful information (Dillon & Goldstein, 1984). The benefit of choosing to use PCA is, when adequately determined, the principal components orthogonally capture the information in new variables which summarize the original ones, simplifies the analysis, and provides additional insight to the data. In PCA, information is in the form of total variance and how it is orthogonally dispersed in the components.

Mathematically, the components are designed to take on as much sample variance as possible; each component is in fact an eigenvector of the correlation matrix. The components are ranked (indexed) according to the size of their corresponding eigenvalues. In this paper, PCA results from only the correlation matrix are used. The rationale behind this decision is given in Chapter III.

1.3. An Example--Determining Which Component Loadings Are Relevant

Loading refers to the scaling of the original variables to that of the component structure. Obtaining the p x p loadings matrix is one of the first step taken after the components are calculated. The elements of this matrix show each individual correlation of the variables to the components. The loadings matrix will be used to draw further inferences regarding the nature of the components. Let's take a look at an example.

In Table 1.1, we see a 6 x 6 loadings matrix with row and column identifiers. The source of data is a study of pilots and engineers taken by groups on several motor and sensory tests (Bauer, 2012). In the first column (on the left side of the matrix) are the names of the original study variables: 'Intelligence', 'Form Relations', etc. The corresponding eigenvalues (λ) ranking of the components for each component is shown in Row 2. Moving to the right along Row 2, the second column is the value 1.7751 for C_1 (component one), the third column is 1.3544 for C_2 (component two) and so on. Each value in the body of the matrix field represents a loading of correlation coefficients between the variables and the components. For instance, the loading for 'Dottings' and C_1 equals -0.7239.

Table 1.1. Component loadings matrix of six variables.

Data Size (40x6)	*Correlation Matrix Derived Eigenvalues*					
	$\lambda_1 = 1.7751$	$\lambda_2 = 1.3544$	$\lambda_3 = 1.0727$	$\lambda_4 = 0.8148$	$\lambda_5 = 0.5306$	$\lambda_6 = 0.4524$
Variable Name	*Component Number (C_i)*					
	C_1	C_2	C_3	C_4	C_5	C_6
Intelligence	**-0.5361**	-0.4614	-0.4783	0.3546	-0.1532	-0.3489
Form Relations	0.1294	**-0.8696**	0.1816	0.1188	-0.2041	0.3719
Dynamometer	**-0.5135**	0.2539	0.4484	**0.6479**	0.1883	0.1248
Dottings	**-0.7239**	0.3660	0.1103	-0.2215	**-0.5148**	0.1258
Sensory	0.4155	0.4142	**-0.6492**	0.3604	-0.1466	0.2879
Perseverance	**-0.7145**	-0.1237	-0.4198	-0.2761	0.3789	0.2795

Still within Table 1.1, under each of the columns for C_1, C_2, C_3, C_4, and C_5 are a number of bold and/or underlined values. Bold font indicates a loading strength between [0.5, 1] or [-1, 0.5] and underlined values are the largest loading for a particular variable. Notice that C_1 has four such loadings under it corresponding to 'Intelligence', 'Dynamometer', 'Dottings', and 'Perseverance'. Moving to C_2, there is only one such loading ('Form

Relations') and similar patterns emerge for C_3, C_4, and C_5. We observe that C_6 has no bold values in its column, indicating it has no strong correlation to any variable.

The analysis thus far appears mundane. If we take a closer look, this time from a perspective of the variables, we see that four variables load to one component but the remaining two, 'Dynamometer' and 'Dottings', each load under two components. Perhaps the level of correlation lends insight into what is going on but we have no such luck. Instead, we are led to the observation that 'Dynamometer' is moderately negatively correlated to C_1 (-0.5135) and stronger positively correlated (0.6479) to C_4. The pattern for 'Dottings' is equally confounding; it is strongly negatively correlated to C_1 (-0.7239) and lesser so to C_5 (-0.6492). We desire good summarization power in the components but determining how important C_1 and C_5 are to 'Dottings' is not clear. It is incorrect to assume we will take the larger loadings value of the two components because this approach violates our intention of explaining maximum total variance. In the case of 'Dynamometer', we would lose ((1.775-0.8148)/6)100 = 16% variance in summarizing power if we adopt this strategy.

We are seeing these artifacts because the component structure is orthogonal and each component will explain (or assume) as much of the *total variance* as possible. The variance assumption process by the components is mutually exclusive and collectively exhaustive: The first component explains the largest proportion of total variance, the second component assumes the next highest proportion, and so on until the last component explains the remaining amount. As such, the structure expands to fill the 'variance space' provided but not all components assume an equal share (the exception

being a perfectly orthogonal set of study variables which yield a correlation matrix equal to the identity matrix). It is justified to be suspect of components with lower ranked eigenvalues that load just one variable, especially if the variable has loaded to an earlier component of a higher rank. In these cases, loadings values *should not* be the primary discriminator for principal component selection. It can be shown that such cases represent mathematically true results with no practical interpretation. We are, in some form, being misled by this sort of variable-component relationship. *What we need is a way to determine the significance of the components*. This situation is indicative of the type problem this thesis seeks to find a solution for.

1.4. Impact of Keeping The Wrong Number of Components: Part I

In the pilots and engineers example, we observed a myriad of difficulties in pin pointing the loadings relevance. We can distill all the cases to just two scenarios: Too few or too many components.

If too few components are kept, we lose summarizing power because we discard some proportion of the total variance. Because the variance dispersion among components is relative to the number of variables in the study, we would omit a nominal amount of variance if there are hundreds of components available and we choose to discard many dozens of the smallest components. However, if just a few variables are in the data, omission of one or two components from further analysis may result in significant loss of information. We may also find the instance of one or two components that explain most or all of the variables. While such an act could be considered a feat of summarization, we are likely more interested in the composite aspect of the component;

that is, what hidden feature of the variables does the component explain?

Conversely, if too many components are kept then we have the case of the 'phantom' loading entry; a single, strong correlation that has no practical significance. In his paper *Stopping Rules in Principal Components Analysis: A Comparison of Heuristical and Statistical Approaches*, Jackson states that one variable significantly loaded to one component "...is not a satisfactory multivariate summary" (1993:2207). In the case of one-variable-to-one-component, we should consider the summarization has been effectively watered down; why retain components that add no insight?

Therefore, given the considerations that too few or too many components is problematic, our goal should be "...to find *the* solution, or at least a solution that others will regard quite highly if not the best." (Horn & Engstrom, 1979:283)

1.4.1. Factor Fission

Factor fission occurs when too many components are extracted, causing loadings to shift suddenly from lower dimension components to higher dimension ones. While factor fission does not occur in PCA, it is a concern for factor analysis (FA). It is mentioned here solely for completeness in regard to the need for an accurate assessment of component dimensionality. As has been pointed out in literature, PCA and FA share similar methodology and both make use of the same stopping rules (Velicer, 1976:324) (Franklin, Gibson, Robertson, Pohlmann, & Fralish, 1995:99) (Zwick & Velicer, 1986:433). The interested reader is directed to the paper *Factor Analysis: Limitations and Alternatives* by Ehrenberg & Goodhardt (1976). They provide an excellent example and thorough analysis of a real-world study in which factor fission occurs. Cattell

(1966:245-247) also gives discussion to factor fission and the number of factors to keep.

This paper does not explore further other multivariate analysis methods nor the factor fission phenomenon. The solutions presented by this thesis were evaluated using PCA methods only; however, by the preceding remarks noted these solutions may be extended as appropriate to factor analytic techniques.

1.5. Component Extraction Stopping Rules

As we have seen, determining the number of principal components is not always a straight-forward process. Fortunately, sound guidance exists to help with this task in the form of component extraction stopping rules. Many such rules exist; however, we shall see that not all of them perform to the same level of accuracy – there is usually an inverse relationship between the accuracy of a rule and how easy it is to apply (that is, simple rules trade accuracy in terms of ease of use and vice versa).

1.6. Analyst Subjectivity

The complex nature of human behavior has not yet been broached. This is not to say analysts play favorites in reaching conclusions, only that varied personal experiences, knowledge, and research goals exist in carrying out a study and interpreting the results. An ideal methodology to determine the number of principal components to retain should minimize subjective evaluation.

1.7. Problem Statement

The need exists for easy access to an accurate visual analysis component extraction stopping rule. A worthwhile endeavor is to design the solution so that it minimizes the amount of interpretation on behalf of the analyst yet leaves enough latitude

for unique conditions or circumstances that exist for all research projects (that is, it does not tie the hands of the analyst by offering too specific or restrictive results).

1.8. Research Objectives

The primary objective of this research is to develop an accurate tool determine the number of principal components to retain when conducting principal component analysis. The strategy to achieve this objective is to survey published literature for visual analysis stopping rule candidates, select an appropriate candidate, and automate (i.e., create a computer algorithm of) the candidate rule to provide the user/analyst with both a computer program input interface for data selection and a visual output providing a synopsis of information captured by the principal components in both graphical and tabulated format. Rather than develop new theory, this thesis uses existing theory to develop this new analytical capability.

The secondary objective is to create a parsimonious solution, which minimizes the size and complexity of the analytical tool created. Once a fully functional algorithm is created, the algorithm will be refined to enhance user-friendliness, provide guidelines to interpret the output, and reduce the data footprint required to run the algorithm.

The intended users of the program are practitioners who need to perform PCA, have access to a computer running MATLAB® (© 1994-2013 The MathWorks, Inc.) Version 7 (Release 14) or greater, and understand their data enough to format it properly for the algorithm. Users require only limited MATLAB skills to enjoy the benefits achieved by employing the automated rule. Finally, any instructions or interface with the user will be free of unnecessary and unclear jargon.

1.9. Key Concepts

Before proceeding, key concepts should be introduced. They are used throughout the thesis and being familiar with them will orient later discussion.

1.9.1. Sampled vs. Random Data

Sampled data are gathered from a real-world experiment. Random data, on the other hand, come from a carefully structured simulation model meant to mimic or make use of an underlying probability distribution of the system being observed. Random data does not exist outside the computer. The methodology presented makes extensive use of both kinds of data. To be distinct in usage, the data origin will be identified.

1.9.2. Use of Ambiguous Terms

Within this thesis, the words ‘factor’ and ‘component’, ‘variable’ and ‘feature’, and ‘observation’ and ‘sample’ are considered synonymous, respectively. The author takes no argument with purists and only strives to be flexible in the chosen vocabulary.

1.10. Assumptions/Limitations

All software have design limitations and understanding not only how to use the software but what goes on (within reason) inside the ‘black box’ is worthy advice. The MATLAB software package was used to develop the solutions for this thesis and where possible, built-in functions (those that are provided as part of the licensed MATLAB library) are used. There are two reasons for this. (1) These functions are generally optimized for speed and accuracy and (2) built-in functions simplify script structure, streamline logic, and cut-down on debugging and troubleshooting. The built-in functions needed to code all techniques for this project have been vetted by the author during

multivariate analysis coursework (Bauer, 2012). The vetting process consisted of coding individually each matrix or matrix function and then comparing the results of similar MATLAB functions *on the same input* to produce identical results.

Because infinite combinations of the number of observations, *n*, and the dimension of the data, *p*, exist there is a limitation to the size of problem the algorithm will be able to solve. A survey of data dimensions found in published analyses was used to determine the dimensionality of data the algorithm can support. Additional technical assumptions and limitations regarding data dimensionality are discussed in Chapter III, Methodology.

1.11. Implications

This thesis does not make the claim of returning an absolute determination of what dimension a particular study is. Such a statement, if it is possible to prove, is not within the scope of this thesis. There is as much art as there is science in coming to a sound conclusion when performing PCA. Consider the outcome of this work as another tool for the multivariate toolbox. In the output analysis summary (full description in Chapter IV, Results and Analysis), an *estimate* of dimensionality is given. A satisfactory PCA assessment is dependent upon other considerations at play from which the analyst must draw forth and distill into a meaningful solution.

1.12. Notation

Table 1.2 shows how variables, notation, and symbols are used in this thesis. Attention was given to use to common statistical terminology. Cells containing " - " signify meaning or usage has no amplifying information.

Table 1.2. Symbols and their meanings used in this thesis.

Symbol or Abbreviation	Meaning	Usage
2OM	Second-Order Model	The linear regression least-squares fitted solution algorithm for estimating Horn's curve for random data using a modified form of the lookup table **T**.
n	-	The number of observations in a dataset; The number of rows in a matrix.
p	-	The number of variables in a dataset; The number of columns in a matrix.
n x p	n by p	Size of a matrix.
(p, n)	Point (p, n)	Two dimensional placeholder representation of a data matrix in Cartesian plane coordinate space; a point mapped into the lookup tables.
(p', n')	User-supplied parameters	Pronounced "p prime, n prime" it is the point-of-interest given as input to the solution algorithms.
C_i	-	Component number or index.
i.i.d.	-	Independent and identically distributed.
Inf	Infinite	Arithmetic representation of values too large to represent in conventional floating-point format such as division by zero (The MathWorks, 2012).
K1	Kaiser's Criterion	A component extraction rule: keep all eigenvalues ≥ 1.0. On graphs, occurs as a straight line at y = 1.0 running the width of the horizontal axis
MEV	Mean Eigenvalue	Refers to the results of completing Horn's algorithm for random data. Usage is not applicable to scree line results derived from sampled (real-world) datasets.
$n^{(-)}$, $n^{(+)}$	-	Lower, upper (respectively) NN variables for n chosen so that $n^{(-)} \le n' \le n^{(+)}$ are closest.
NaN	Not a number	A data type that results from operations having undefined numerical results (The MathWorks, 2012).
NN	Upper & Lower Nearest Neighbor	Given a (p', n'), search **T** for values of p and n immediately adjacent to the point-of-interest. Concept extends to **S** and **Y**.
$p^{(-)}$, $p^{(+)}$	-	Lower, upper (respectively) NN variables for p chosen so that $p^{(-)} \le p' \le p^{(+)}$ are closest.
$\mathbf{R}_{(p \times p)}$	-	Correlation matrix of p rows by p columns.
ROI	Region-of-Interest	The 2D area of mapped (preprocessed) MEV data for select sizes and intervals of (p, n).

T	-	A user-defined total variance target to be explained by full or partial selection of component eigenvalues.
S	Row and column truncated **T**	Rows containing $p^{(-)}$, $p^{(+)}$ information used for nearest neighbor search refinement (child of **T**)
T	Lookup table matrix	Shorthand reference to the matrix storing the mapped ROI data. In Appendix II, the actual lookup table variable name is `tablex` (MEV) or `tablexbeta` (2OM).
$\mathbf{X}_{(n \times p)}$	-	Data matrix of n rows by p columns. - In random data, **X** has random variables ~NID(0,1). - In sampled data, **X** has empirical elements.
Y	Row truncated **S**	A four row, subset matrix of **S** used for the surrogate curves interpolation routine (grandchild of **T**).
x_i	-	A variable in a linear regression model.
x_{ij}	-	A matrix data element at row i and column j.
β_i	Lowercase (LC) beta	Regression coefficient; subscript i indicates order of the coefficient.
$\hat{\boldsymbol{\beta}}$	LC beta hat	Matrix of estimated regression coefficients in the 2OM.
λ	LC lambda	Eigenvalue; an element of the hidden component structure.
$\bar{\lambda}_i$	LC lambda bar	Pronounced ""; indicates the arithmetic mean of the eigenvalue at index i.
$\hat{\lambda}_i$	LC lambda hat	Pronounced "lambda hat"; indicates the approximation the linear regression second-order model produces for the eigenvalue at index i (λ_i)
$\sum_{i=1}^{p} x_i$	Uppercase sigma	Generic summation of elements x_i from 1 to p.
~	Approximately	Close to but not exact in value (relational).
~NID(μ,σ^2)	-	Normally and independently distributed of parameters mean μ and variance σ^2.
$\leq$	Less than or equal to	-
$\geq$	Greater than or equal to	-
$\pm$	Plus or minus	-
$\equiv$	Identical to	Denotes "is defined as." Not to be confused with equality ("=")

$\neq$	Not equal to	-
`text`	MATLAB origin	A reserved or user-defined command, function, or variable found in a MATLAB script (program). May also identify a script or data filename reference when highlighting adds clarity among body text.
ε	LC epsilon	A small value approximating zero; significantly smaller than others in like comparison; an error or residual (difference between predicted and observed regression model values).
$\mathbb{N}$	The set of natural number	Integers 1, 2, 3, ..., ∞. In the context of this thesis, the set does not include zero. To reinforce the restriction, $0 \notin \mathbb{N}$ is specified wherever $\mathbb{N}$ is used.

1.13. Chapter Summary

PCA is a multivariate analysis technique used to summarize total variance of a dataset through discovery of hidden component structure. This makes analysis more tractable as not all variables need to be retained for further analysis; the summarization of the components enables us to find new dimensions in which to express the data. We learned that components are formed from linear combinations of the original variables. Each component represents an eigenvector and is ranked by the magnitude of the corresponding eigenvalue. The orthogonal design of the components is such that the first principal component assumes as much variance as possible, the second assumes as large a portion of the remaining share as it can, and so on until the last component accounts for the remaining small fraction.

We looked at an example of a loadings matrix which is a scaling (or correlation) of the original variables with the components. We saw that the loadings matrix can be unclear to decipher and that it is possible for too few or too many components to be included in the analysis. Hence, there is a need to apply a stopping rule so that one

knows when the estimated dimensionality of the components has been determined.

Finally, key terms and definitions of principal components analysis were introduced. The research objective is to review the literature for an accurate existing visual analysis component extraction stopping rule and automate it. Automation will help minimize analyst subjectivity during PCA. The finished algorithm should present an easy-to-use interface for the analyst and provide an output analysis product with relevant summary information.

II. Literature Review

2.1. Historical Perspectives

The purpose of this chapter is to provide relevant background information regarding principal components visual analysis stopping rules. Because the methods discussed are decades old yet still used largely as first developed, understanding the scope of problems they were invented to solve and limitations in application provides context to carry forward to later sections of the thesis.

Three such rules were found in the published literature. The reader may find the discussion on test results of these various techniques enlightening – the findings of three different papers under at least as many authors are shared – and the perils of dimensionality assessments done poorly.

Lastly, we examine what a realistic expectation is for how large a multivariate problem can be evaluated. We do this by surveying the sizes of data in published researched or of datasets posted to public access websites.

2.1.1. A Note About Verbiage

Because the literature review is a walk through history, the vocabulary encountered is a mix of old and new. Where possible, explanatory and supporting graphics make use of the author's word choices and, if available, the size of data used in the article. Doing so not only makes it easier to relate a figure to the story in the literature, but the authors, some of whom are no longer with us, are given a chance to share their ideas again. Understanding how thoughts and concepts were developed leads to deeper knowledge of the solutions offered. Therefore, the reader should expect to see

apparently inconsistent word usage in different parts of the literature review. By the end of this chapter, standard terminology is adopted.

2.2. Graphical PCA Techniques

A good place to start a literature review of principal components analysis (PCA) is with the groundbreaking work done on the topic in the 1960s. The goal of PCA then was the same as it is today: Given a multivariate data set, provide to the researcher a reliable statistical tool to either (1) describe the variance shared among variables in a study or (2) parsimoniously describe the total variance of those variables (Velicer, 1976). The work done in those decades is fundamental to an understanding of how PCA is carried out today. The theory and findings of early researchers remain quite relevant and in use. Therefore, not only is a survey of legacy material justified, it should be done as due diligence because it is in the original papers that the ground breaking authors share the theory, application, limitations, and pitfalls of their accomplishments.

Multiple pioneers worked to build the PCA toolbox and a few names and techniques stand out. Three stopping rules requiring visual analysis are (in chronological order of first publication) Henry Kaiser's eigenvalue > 1.0 criterion, John Horn's curve, and Raymond Cattell's scree plot. All were developed before inexpensive and powerful computers were commonplace.

2.2.1. Scree Line Definition

Fundamental in each rule is the use of a scree line. A scree line is simply a line drawn in Cartesian coordinates (*x*-*y* plane) between each eigenvalue (the *y*-axis or ordinate) as the eigenvalues are plotted over the integer index of the corresponding

component (along the *x*-axis or abscissa). In this thesis, the scree line is always represented by a blue line connected to large orange points when it is drawn in figures. The description 'scree' was chosen (not by this author, but others) because of its similarity to the rubble that falls off a cliff and slides to the bottom of the hill.

2.2.2. Latent Roots

For each of the methods surveyed, a "latent root" is equivalent to a component (eigenvector) and the ordering sequence is determined by magnitude (eigenvalue; λ). Both are results of eigendecomposition of the correlation matrix **R**.

2.2.3. Kaiser's Criterion

This is perhaps the most common stopping rule (Velicer, 1976) because it is the easiest to apply. Found abbreviated as K1 in the literature (Zwick & Velicer, 1986), it states that the number of factors to be retained is equal to the number of latent roots greater than one in the observed correlation matrix (Kaiser, 1960). As a visual element in a scree line graph, Kaiser's criterion appears as a straight line at $\lambda = 1.0$ running the length of the horizontal axis over successive components C_i. The rule is simple: Components above the line are principal, the ones below the line are not. The rationale can be considered from the perspective of what makes an effective executive summary: One would not write a lengthier synopsis than the source is long.

Figure 2.1 shows an example of the rule being applied. Here, components C_1, C_2, and C_3 have eigenvalues greater than one and are above the K1 line – they would be retained as principal components. The rest (C_4 - C_{13}) are below it ($1 \geq \lambda > 0$) and would be discarded as not significant to further principal components analysis.

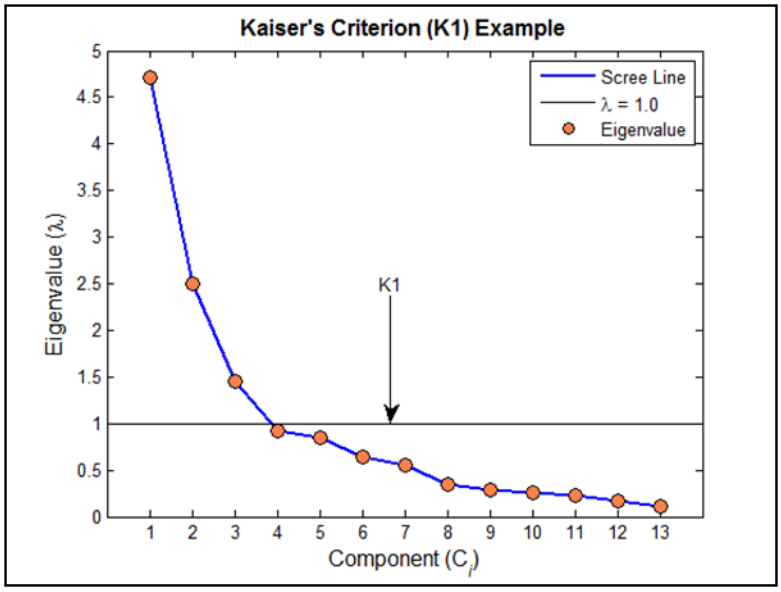


Figure 2.1. Kaiser's criterion is always found at λ = 1.0. Size of data shown is 178x13.

2.2.3.1. Considerations Regarding Kaiser's Criterion

Kaiser's criterion performs in a binary manner; that is, the distinction is either above or below λ = 1.0. There might be subtleties that require closer inspection. Is there a practical significant difference between λ = 1.01 and λ = 0.99? Horn has pointed out K1fails to recognize sampling error due in part to the assumption that K1 operates on population parameters assuming infinite sample size (Horn, 1965:181).

2.2.4. Horn's Test

John L. Horn's paper *A Rationale and Test for The Number of Factors in Factor Analysis* (1965) describes what has come to be known as Horn's test or Horn's procedure. Also called parallel analysis or PA (Zwick & Velicer, 1986) (Franklin, Gibson, Robertson, Pohlmann, & Fralish, 1995), Horn begins with the following theory. If we let *k* be large and if *k* sets of size *n* x *p* are drawn randomly from a population of numbers independently and identically distributed (i.i.d.) according to the normal

probability density function, the *p* x *p* matrix of correlation coefficients **R** will approximate an identity matrix. Within **R** is a set of latent, positively-valued roots and each root accounts for some amount of variance within each *p* inter-correlated variables. As in K1, these latent roots are eigenvectors and are ranked according to the size of the eigenvalues from eigendecomposition of **R**. What is new is Horn's curve is formed from the means of each ranked eigenvalue (amplification of the technical details is given in Chapter III). Should an infinite sample size be considered, all correlation coefficients will equal 1.0. In an actual experiment, however, the researcher must contend with a much smaller sample size and the accompanying sampling error and least-squares

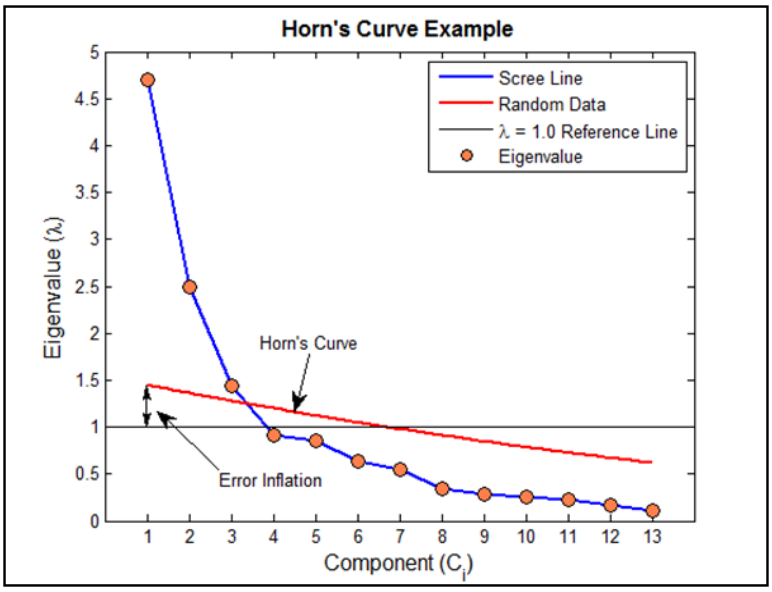


Figure 2.2. Horn's curve example for a sampled dataset. The point p/2 is approximated between C_6 and C_7. Size of data shown is 178x13.

'bias' (1965:180). If we consider these elements as simply 'error,' then we can illustrate how the combination of the two has inflated the correlation in **R** for the first *p*/2 components. In Figure 2.2, the scree line represents the data sample. By comparing its

slope to that of the error induced curvature for the population (red line; Horn's curve), we can measure the difference between the two, adjust the analysis for it, and reach a reasonable conclusion of how much of the variance is due to sampling error.

What Horn proposes is a method to separate signal from noise; in other words, distinguish meaningful information in the data by adjusting for the amount of error expected due to random chance. Therefore, an estimate of the level of noise in a sample provides an indication which components should be considered for extraction. Expressed graphically, Horn's curve estimates the pure error in the sample. Eigenvalues above the curve contain useful information; eigenvalues below the curve (and especially beyond $p/2$) are 'noisy' and should be discarded.

2.2.4.1. Considerations Regarding Horn's Test

Horn's test is not widely used because it requires a large amount of simulated data to be generated for each n x p of interest (Monte Carlo simulation of repeated random draws from a standard normal probability distribution). As such, it carries a data footprint with it in the form of preprocessed data tables or requires lengthy on-the-fly calculations. Monte Carlo simulations for large n x p can take a significant amount of time to complete and if there is another stopping rule available, it is possible practitioners would prefer to use the quicker solution. Because modern computers can make easy work of the techniques Horn describes, this limitation can be mitigated.

2.2.5. Cattell's Scree Plot

Introduced by Raymond B. Cattell (1966) the scree plot begins with a typical scree line and then looks for breaks in the scree line slope. To perform the scree plot test,

the scree line is first drawn. Any abrupt changes (or breaks) in slope along the scree line are noted, particularly the first one to occur (i.e., closest to the vertical axis). Next, the analyst traces a line segment using two points – the first at the last eigenvalue in the sequence (at C_p), the second at the eigenvalue where the break in slope was noted – and

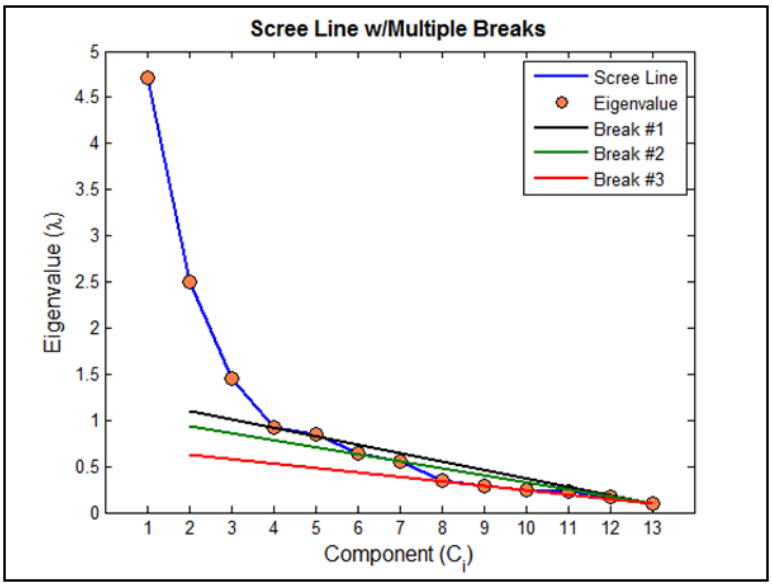


Figure 2.3. Scree plot illustrating three possible breaks in slope: Break #1 retains three roots, Break #2 retains five roots, and Break #3 retains seven roots.

retains the eigenvalues above the traced line and discards the ones below it. Because each eigenvalue represents one corresponding component, by corollary an estimation of dimensionality has been made. Figure 2.3. illustrates the concept; the black, green, and red lines are all drawn in the manner described.

Cattell admits there is an art to the effective use of this technique and the application requires a thorough understanding of the process subtleties (1966:256-261). When using this method, questions to ask are "Is there more than one break indicating multi-modal data?", "How should the inherent changes in line inflection be evaluated?"

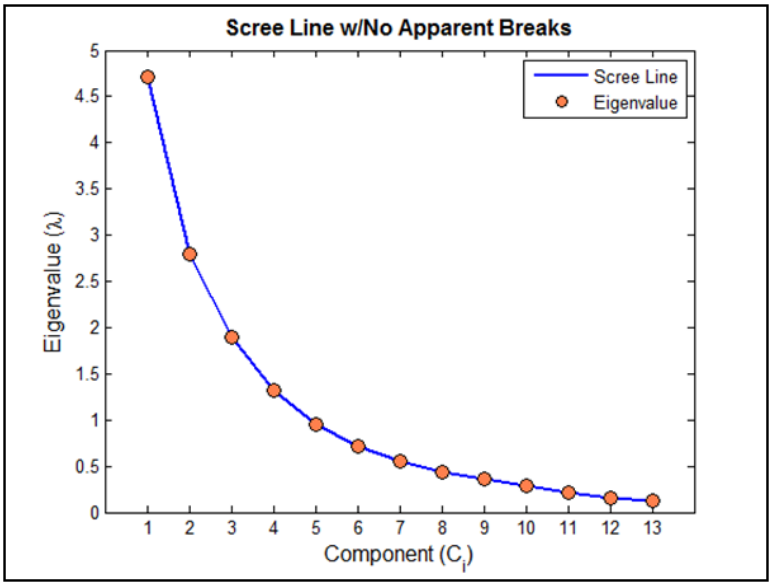


Figure 2.4. Scree line with no apparent breaks in slope.

and "What if there are no apparent breaks in the scree line?"

In concluding his paper, Cattell states (p. 273) "There is no true thing as 'the true number of factors to extract'....Consequently, the cut-off point in extraction is best decided by a conception of non-trivial common variance…" He then provides a series of suggestions to consider, all of which are not applicable to visual analysis techniques so they are excluded from further discussion (pp. 273-274).

2.2.5.1. Considerations Regarding Cattell's Scree Plot

Cattell's method is open to subjective interpretation on part of the user, something we stated we wish to minimize in our candidate stopping rule. In Figure 2.3. we have the case of multiple choices of breaks in the scree lines and in Figure 2.4. we have exactly the opposite--no apparent break in slope at all. However, it is more likely in practice that the situation encountered in Figure 2.3 will be found than of that in Figure 2.4. There is not much ground to be gained by automating the scree plot; however, it will not be ruled

out as a candidate at this time.

2.3. Objective Evaluations of Stopping Rules Accuracies

The next part of the literature review is a survey of three different journal articles, each performing comparative tests of the stopping rules we have examined. Each article tested many different stopping rules; however, our scope is limited to discussion of the findings summarized in Table 2.1 for the three graphical techniques of interest to us.

2.3.1. Summary of Test Findings--Zwick and Velicer

In broad discussion, Zwick and Velicer determined that PA was overall the most accurate method they tested; however, depending on the composition of the test data, PA could perform slightly different with different sample sizes (1986:434).

They did not recommend K1 for PCA as it consistently overestimated the number of major components. Velicer, in an earlier work, makes the statement that Kaiser's greater than unity rule and the scree test both have been criticized as either too subjective or too arbitrary (Velicer, 1976:322). The criticism of K1 a decade later is stronger: "The use of the K1 rule as the default value [in popular statistics software] is an explicit endorsement, particularly to naïve users…seems to guarantee that a large number of incorrect findings will continue to be reported." (Zwick & Velicer, 1986:439). Despite positive aspects of the scree test, they did not recommend it, as the subjectivity in using it invites concerns regarding the practitioner's reliability. However, the scree may be useful as an initial estimate or as a method complementary to PA. The major drawback of using PA is the need to generate large sets of correlation matrices at the particular combination of n x p (Zwick & Velicer, 1986:441).

Their chosen test method generated five sample correlation matrices from 48 known population correlation matrices (at each of two sample sizes). There were six levels of component pattern complexity in the 48 correlation matrices. The approach was similar to the "middle model" of work published by Tucker et al. in 1969 (Zwick & Velicer, 1986:435).

2.3.2. Summary of Test Findings--Jackson

Jackson did not include PA in his analyses; however, under discussion of the scree plot he recognizes Horn's 1965 paper and restates its methods but does not use it in forming a solution (he uses Cattell's 1966 procedure as outlined). Jackson found that K1 tended to overestimate the number of dimensions to retain and the scree plot tended to consistently retain one too many components (1993:2211). Jackson's test data consisted of simulated data matrices of uniform correlation structure, patterned matrices of varying correlation structure, and three ecological-based datasets of lake water samples.

2.3.3. Summary of Test Findings--Peres-Neto et al.

The most recent comparative study surveyed was published by Peres-Neto, Jackson, and Somers (2005). In total, they compared 20 PCA methods, two of which are of interest to this thesis (K1 and PA; the scree plot was not part of the test group). Their results suggest that irrespective of matrix size and type of distribution, PA was one of the most accurate rules overall and called it "...[one of] the most promising rules for component evaluation" (p. 994). K1 performed poorly and was removed from further inspection due to poor performance.

Their chosen test methodology included Monte Carlo protocol produced

correlation matrices in 9 or 18 variables with known non-trivial components. Trivial components were degenerate and carried only noise. In total, fourteen different designs of correlation matrices were used.

2.4. Impact of Keeping The Wrong Number of Components: Part II

In Chapter I we made an observation from the loadings matrix example (Table 1.1) that a need exists to balance summary with clarity. Which is the greater PCA misstep: Retention of excessive trivial principal components (too many components) or including only those that are unambiguously relevant to the analysis (too few components)?

Cattell is of the opinion (1966:246, 275) that allowing some amount of variance (error) into the analysis is acceptable and is even encouraged; this is accomplished by permitting an extra component into the analysis. If the aim of the PCA is exploratory in nature, this approach may be well-suited. Note that Cattell frames his discussion from a factor analysis perspective. Interested readers are encouraged to reference his 1966 paper *The Scree Test for The Number of Factors* paying particular attention to pages 245-247.

Zwick and Velicer give three factor desirability guidelines to consider when determining an experimental goal (1986:432-433). They recommend (1) a component have three significant (non-zero) loadings to be useful; (2) summarizing power greater than 1.0; and (3) non-negative reliability (a reference to a test design statistic). Zwick and Velicer categorize the interest components have to a researcher as:

- Major components have three or more substantial loadings and are probably of interest;

- Minor components have less than three substantial loadings but an eigenvalue greater than 1.0 or an eigenvalue less than 1.0 but with three or more substantial loadings are probably of interest;
- Trivial components have an eigenvalue less than 1.0 and less than three substantial loadings should not be retained.

Peres-Neto et al. investigated the matter and found that over extraction might not be as serious a problem as under extraction (2005:994), the reason being earlier components (those with larger eigenvalues) have higher amounts of variance. Thus, if too many components are kept, one is likely to be retaining a small amount excess variance instead of cutting out a large amount of useful information by under extracting.

2.5. Summary of Component Extraction Stopping Rules

We now have the information necessary to select a stopping rule candidate. Based upon the observations and findings of the literature review, this author's collective thoughts of what he has read and of what others have published and a summary of the component extraction stopping rules accuracy tests, the following criteria will be used to select a component extraction stopping methodology:

- Be a visual analysis method;
- Reduce unnecessary subjectivity on behalf of the analyst; and
- Produce accurate dimensionality estimates.

Table 2.1 is a chart of the benefits and limitations of the stopping rules, how they compared to each other in independent testing, and an overall assessment. From it we can make a conclusion regarding which candidate to select for development.

Table 2.1. Composite score decision matrix for stopping rule selection.

		Test Method Candidates		
		Kaiser's Criterion (K1)	**Horn's Method (Parallel Analysis)**	**Cattell's Scree Plot**
Literature Review Findings	**Pro**	Easiest to apply, very popular	Accounts for error in the sample	Flexible; analyst can make choices
	Con	Inflexible due to the go/no-go results	Requires large amount of random data for each *n* x *p*	Possibly misleading if scree line has complex slope
Comparative Tests Findings	**Zwick & Velicer (1986)**	Not recommended for PCA	Recommended; regarded as most accurate	Recommended; esp. for experienced investigators
	Jackson (1993)	Overestimates non-trivial dimensions	(Not evaluated)	Overestimated interpretable components
	Peres-Neto et al. (2005)	Removed from further testing due to poor performance	One of the most accurate overall	(Not evaluated)
Assessment		Nothing to gain by automating	Software can provide demand for data 'overhead'	Would require practitioner training, experience
Conclusion		**Not selected**	**Selected**	**Not selected**

Of the stopping rules surveyed, Horn's method offers the advantages of accurate estimation, is not prone to subjective analysis, and has familiar key features of the other methods (e.g., the scree line and reference to $\lambda = 1.0$). Having this quality (i.e., familiar features) was not part of the original research objective, but knowing now that K1 – despite its misgivings – is a popular rule, incorporating the $\lambda = 1.0$ element can provide an additional benchmark without compromising the automation goal.

In order to make effective use of time processing random data for Horn's test, we need to first determine what sizes of data are found in published works. The size and number of problems the algorithm can answer will be limited. Preliminary surveys (the

technical challenge will be left for discussion in Chapter III) provide insight on just how large a dataset a 'typical dataset' might be.

2.6. Expected *n* x *p* of Research

While more data is usually a good thing, at some point too much becomes overwhelming. It is not a research objective to provide Horn's curve for every possible experiment because there are infinite combinations of *n* x *p*. Instead, the effort is to work smarter, not harder, and frame the *n* x *p* solution in terms of a region-of-interest (ROI); a two-dimensional area defining practical and relevant bounds of both *n* and *p*.

During the literature review, each article and web search using a dataset of stated *n* observations and *p* variables was recorded. (Note: Not all literature revealed their sample sizes and member variables; some articles stated only summary findings.) To adequately define the ROI, actual data was not of import, only the *n* x *p* dimensions. The goal is to gain understanding of what data sizes are expected in the research community. Of particular benefit was the Machine Learning Repository (University of California-Irvine, 2007). The UCI website is replete with donated multivariate data and organizes its database by field of research, data sizes, year the research was conducted, and purpose (classification, regression, etc.). To narrow the problem scope, multivariate listings that had at least one publication citation were logged. Using this criterion, 156 such datasets were recorded. Within the published literature, 22 instances of listed data sizes were found. Altogether, this represents 178 'samples' of empirical research; Figure 2.5 shows how they are clustered.

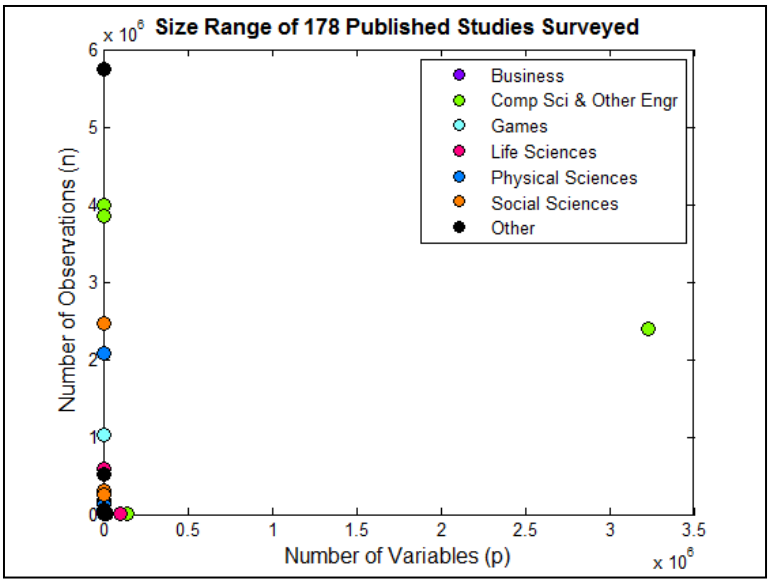


Figure 2.5. Data sizes of the surveyed, published studies. Axes scales are in hundreds of thousands.

The area in Figure 2.5 is too broad a range of *n* x *p* to work in; the amount of sparsity present suggests that mega-sized datasets are not the norm. Note that the green points, representing the computer science field, are very large in one or both dimensions and represent most of the outliers. These datasets contain information such as web page visits, online user surveys, and optical recognition of character symbols. It is not unexpected that automated data collection routines and the sheer volume of Internet activity results in such large datasets.

To reduce the outlier clutter, filtering is done on studies that number fewer than 10,000 observations. One hundred and forty three studies are in this range (Figure 2.6). A reasonable portion from which to form the ROI appears in the red bordered area; this area encompasses studies having 1 to 1,000 variables *p* and 1 to 7,000 observations *n*. Defining the ROI in this manner captures 80.3% of the original 178 *n* x *p* 'samples.'

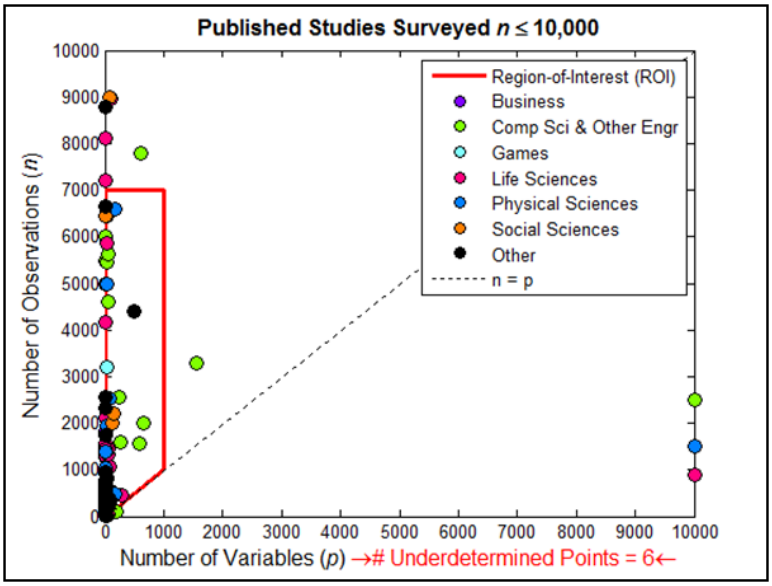


Figure 2.6. ROI bordered in red. Note clustered studies near the origin.

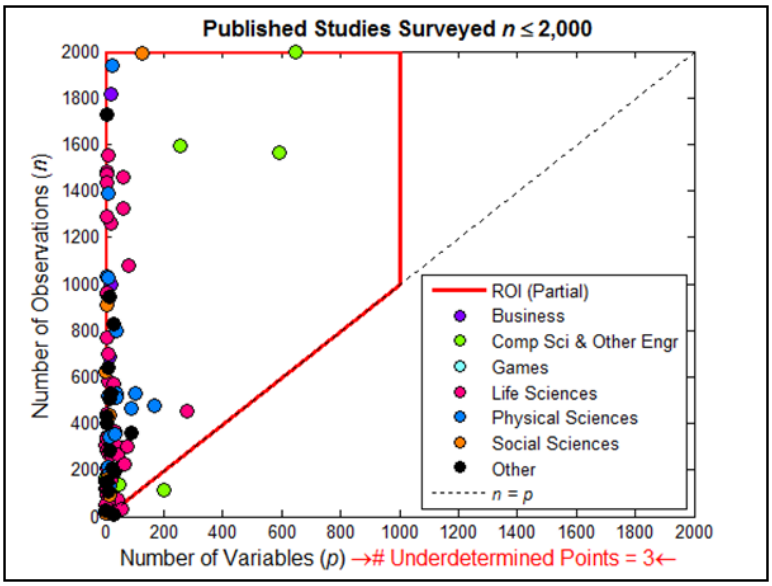


Figure 2.7. Magnified view of studies clustered near the origin. Three points are underdetermined (below the diagonal). Red ROI border is omitted for clarity.

In Figure 2.7 a magnified view of the lower part of the ROI is presented; here we can see the individual elements (studies) and how the pattern appears to be most published multivariate work contains fewer than ~150 variables and 500 or so instances. The number of studies in this group is 111 (62.4%).

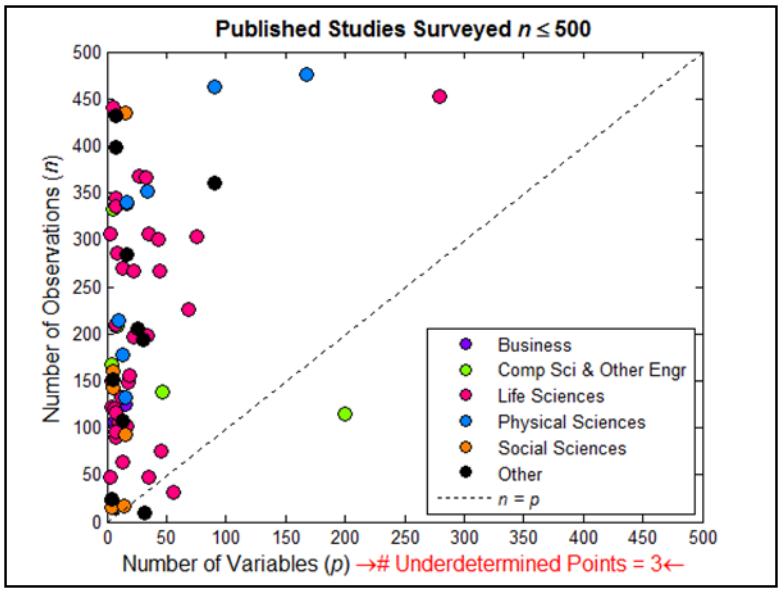


Figure 2.8. Close-up view of the origin. Three points are underdetermined (below the diagonal). Red ROI border is omitted for clarity

The final snapshot of the ROI is taken by scaling the sample size down to $n \leq 500$. Here we have a satisfactory view of the grouping near the origin at (0, 0). As shown in Figure 2.8, the number of studies within this range are 75 (42.1% of the 178). At this resolution there is enough detail to see that studies having *underdetermined data* are rare, accounting for only 3 of 75 (0.4%) of the works noted. The matter of underdetermined data (more variables than observations in a study) will be revisited in detail during Chapter III, Section 3.7. For now, it is sufficient to say underdetermined data presents technical challenges and all such datasets will be excluded from analysis.

2.7. Chapter Summary

In this chapter the literature was reviewed for visual analysis component extraction stopping rules. Three candidates were found:

- Kaiser's criterion (K1);
- Cattell's scree plot; and
- Horn's test.

Each test has its own drawback design limitation. In particular, Horn's test requires a large amount of random data evaluation before it can provide a dimensionality estimate for a sampled data set. Graphically, Horn's procedure appears to incorporate features of both the scree plot and K1.

To gauge how well each candidate performs in determining the number of principal components, three published articles on comparative evaluation of stopping rules were reviewed. These papers put each stopping rule under test using carefully constructed simulation data of which the dimensionality was predetermined. Horn's test received high marks for being one of the most accurate methods tested. K1 tended to overestimate the number of principal components (that is, 'noisy' components with little substantive value were included in the dimensionality determination). The scree plot was accurate in its dimensionality estimates as long as the practitioner was experienced with the technique. The summary observations, findings, and candidate selections are listed in Table 2.1.

We approached the question of how many factors to retain once more (first presented in Chapter I), this time reviewing what other authors had found. Most stated it

is better to include too many components rather than too few, as leaving components out may discard useful information (i.e., variance) in the eigenvalues.

At the end of the stopping rules survey, it was apparent that Horn's test is the smart choice for algorithm development and has been selected as our candidate. The data requirements needed to run the routine had to be addressed, so a survey of the published literature (both printed and found in the UCI Internet database) was conducted to determine what sizes of n x p are likely to be found in experimental studies. The ROI was identified and the data processing budget was centered on an area encompassing roughly 1 to 1,000 variables p and 1 to 7,000 observations n. We want the algorithm to be of practical utility to multivariate practitioners and mapping of the ROI will build-in value.

III. Methodology

3.1. Chapter Overview

This chapter discusses the methodology used for completion of this thesis. Starting with Horn's ideas in *A Rationale and Test for The Number of Factors in Factor Analysis* (1965), we develop the ideas and thought processes and move toward a functioning MATLAB algorithm.

We start the journey with the thought we have some sense of direction but are not completely sure what we will find. Horn gave us all the parts of how and why the procedure works. The exact process – the writing of the algorithm – does not contain any surprises but it does require some careful thought about how to set the stage. These next few paragraphs will discuss the integration of the parts to a whole and why decisions have the outcomes that they do. Since we are exploring, taking time to visually see what is happening in the data is often more revealing than staring at a column of numbers – visual evidence is often quite compelling. The theory and rationale as a whole are first introduced and as we progress, supporting concepts are visited:

- Monte Carlo simulation (MCS);
- Scree lines characteristics;
- Refinement of the region-of-interest (ROI);
- Lookup table characteristics;
- Linear interpolation and nearest neighbors (NN) search;
- Linear regression second-order model; and
- Sampled data algorithm.

At chapter end preliminary results will be demonstrated as well as side-by-side comparisons of the two solution approaches (a mean eigenvalue routine and a linear regression second-order model).

3.1.1. Mean Eigenvalue (MEV) Solution

The first solution strategy is a pre-processed table of data capable of producing an on-the-fly estimation of Horn's curve. In Section 2.6 we observed that not every possible *n* x *p* combination is of interest to us. Therefore, an interpolating function is needed to find intermediate *n* x *p* solutions that fall within the ROI but for which we do not have specific information for. MEV is the most direct route to a solution but it is the most burdened with the data overhead it requires. The MEV solution will be developed first.

3.1.2. Linear Regression Second-Order Model (2OM) Solution

The second approach is to build a 2OM model on the framework of the MEV solution. Depending on the model coefficients, a second-order polynomial can produce a curve ranging from a parabola to a nearly straight line (we assume the quadratic coefficient is not zero; otherwise, we would choose a first-order model). We maintain the MEV constraints on *n* x *p* data selection plus the requirement that Horn's curve be 'well-behaved' (i.e., it has predictable properties for all points within the ROI). The response function that captures this requirement is strictly monotonically decreasing.

3.2. Motivation

As was shown in the literature review, Horn's technique is generally regarded as producing an accurate assessment of component dimensionality. Its drawback is the need to generate large amounts of random, normally distributed data. In the decades since

Horn first described the technique, advances in computer resources have made it possible to automate a fast running algorithm and computer memory and hard drive storage is abundant. The impact of a large data footprint can be minimized.

3.3. Theory of Horn's Test

We begin by letting n = sample size, p = number of variables, and k = a large number of Monte Carlo simulation (MCS) iterations. The random variable distribution is configured standard normal (population parameters mean $\mu = 0$ and variance $\sigma^2 = 1$) within the MCS. If $\mathbf{I}_p$ is the p x p identity matrix, the individual elements x_{ij} of random data matrix $\mathbf{X}$ are

$$x_{ij} \sim NID\left(0,\ \mathbf{I}_P\right) \text{ where } \begin{cases} i \in \{1, 2, ..., n-1, n\} \\ j \in \{1, 2, ..., p-1, p\} \end{cases} \tag{3.1}$$

Upon each increment of k, a new data matrix $\mathbf{X}$ of random variables is produced. It should be emphasized that *each element* in $\mathbf{X}$ is created i.i.d. during *each iteration*. There is no recycling of data between iterations.

3.3.1. Correlation of The Random Data

The correlation operator on $\mathbf{X}_{n\times p}$ is consistent with Horn's methodology (1965:179-182) and by evaluating correlation results scaling difficulties and nonsensical units of measure are eliminated. The eigenvalues λ_i are extracted from the resultant correlation matrix $\mathbf{R}_{p\times p}$ by determining the linear combination $\sum_{i=1}^{p} a_i C_i = 0$ of the variables $C_1, C_2, ..., C_{p-1}, C_p$ having maximum sample variance, rank-ordered from largest to smallest (by index i: i = 1, 2, …, p-1, p), and then averaged by index. In notation, the

steps are:

$$\mathbf{X}_{n\times p} \rightarrow \text{Correlation operations on } \mathbf{X} \Rightarrow \mathbf{R}_{p\times p} \tag{3.2}$$

$$\mathbf{R}_{p\times p} \rightarrow \text{Eigenvalue extraction and sorting } \Rightarrow \lambda_1 > \lambda_2 > \cdots > \lambda_{p-1} > \lambda_p \tag{3.3}$$

$$\sum_{j=1}^{k} \lambda_i \rightarrow \text{Sum within each sorted index } i \tag{3.4}$$

and finally the average eigenvalues are sorted and summed by index

$$MEV \equiv \bar{\boldsymbol{\lambda}} = \frac{1}{k}\left[\sum_{j=1}^{k}\lambda_1, \sum_{j=1}^{k}\lambda_2, \cdots, \sum_{j=1}^{k}\lambda_{p-1}, \sum_{j=1}^{k}\lambda_p\right] \tag{3.5}$$

and stored in vector format for further use. The symbol $\bar{\lambda}_i$ ("lambda bar sub *i*") denotes the MEV for component *i* where $i \in \{1,2,...,p\text{-}1, p\}$. Vector $\bar{\boldsymbol{\lambda}}$, Equation (3.5), contains all the ordinate information for completing Horn's curve.

By rank-ordering the eigenvalues from highest to lowest, summing within each rank (index), and then averaging the indexes, a picture of how much total, or common, variance each eigenvector represents emerges. A natural next step is to use the mathematical properties of the components to study the behavior of the original variables expressed in new combinations of each other.

In contrast to sampling theory not all eigenvalues are equal in the real-world but they all sum to the number of variables in $\mathbf{R}_{p\times p}$.

$$\sum_{i=1}^{p} \lambda_i = \lambda_1 + \lambda_2 + \lambda_3 + ... + \lambda_{p-1} + \lambda_p = p \tag{3.6}$$

This is an interesting property regarding eigenvalues of the correlation matrix – a total of *p* eigenvalues, equal to the number of columns (each column representing one variable)

in **R** – collectively sum to *p*. In PCA, we take advantage of this relationship in two ways. One, it is straightforward to determine the fraction of total variance the i^{th} component explains from the equation

$$\frac{\lambda_i}{p},\ i \le p : i, p \in \mathbb{N}\ (0 \notin \mathbb{N}) \tag{3.7}$$

By extension, we reach advantage two: A target variance *T* can be chosen by the practitioner and approximated by selecting a full or partial sum of *j* eigenvalues

$$\frac{1}{p}\sum_{i=1}^{j}\lambda_i \cong T \begin{cases} j \le p : i, j, p \in \mathbb{N}\ (0 \notin \mathbb{N}) \\ 0 \le T \le 1 \end{cases} \tag{3.8}$$

Note that while *T* may be any value, the number of eigenvalues is discrete so an exact summation to *T* is not likely; *T* is more properly used as a threshold. The art of target variance application should be done before the dimensionality assessment. How much variance the researcher wishes to summarize should be kept in sight of determining how many principal components to retain as the two are directly linked.

3.3.2. Covariance Instead of Correlation

PCA can be performed using the covariance operator; however, the sample variables should be similar sized and of consistent measurement categories. As was previously stated, choosing to work with the correlation matrix eliminates unusual units of measure that may not be apparent without dimensional analysis of the variables. Hence, the algorithm born in this thesis is applicable *only* to summary statistics from correlation operations. Readers proficient in MATLAB coding and who require the use of variance-covariance operations are encouraged to make the necessary modifications to the algorithm (located in Appendix II).

3.4. Monte Carlo Simulation

In his paper, Horn mentions "...sets of very large samples of size...drawn independently from a normally distributed population of random numbers..." and "...Insofar as [the iterations] are reasonably large, these averages give R_a [Figure 3.4; the curve of random data for large observations]." (1965:179; 182). An appropriate routine to the large random samples requirement is Monte Carlo simulation (MCS), the technique of using repeated sampling to determine properties or behavior of some phenomenon. Formally, Sawilowsky (2003:219), refers to MCS as "... an explicit reference to the use of repetition as a method of discovery of the long run outcome of an event."

Given that the parameter of interest θ is the magnitude of each rank-ordered component (the mean eigenvalues $\bar{\lambda}_i$ where $i = 1, 2, ..., p$), we let $\mathbf{X}$ be a discrete random vector and the parameter of interest the some specified function h is

$$\theta = E[h(\mathbf{X})] = \sum_{j=1}^{\infty} h(x_j) P\{\mathbf{X} = x_j\} \tag{3.9}$$

When $h(x_j)$ is difficult to evaluate, the use of random numbers can be used to generate a partial sequence of i.i.d. random vectors $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n$ having the mass function $P\{\mathbf{X} = x_j\}$, $j \geq 1$ (Ross, 2007:247-248). The strong law of large numbers yields

$$\lim_{k \to \infty} \sum_{i=1}^{k} \frac{h(\mathbf{X}_i)}{k} = \theta \tag{3.10}$$

Therefore, for large k the average approximates $\bar{\lambda}$. The strong law of large numbers guarantees the approximation to the parameter as k, the number of iterations, becomes large. Further details regarding choosing k are discussed in Sections 3.4.2 and 3.12.2.

3.4.1. Random Number Generator

MATLAB contains varied (pseudo)random number generator functions (PRNG) and user-adjustable parameters. MATLAB version 7.12.0.0635 (release R2011a) was used for all scripting. Prior to Version 7.7, the seeding of the random number generator was somewhat confusing and at risk of being misapplied (The MathWorks, 2012). Because of this, the `rng` function was introduced. By setting `rng(z)`, where `z` is non-negative integer, the programmer controls the stream used by the random number generator functions – these functions include the `normrnd` call shown in Figure 3.3.

In all instances, the PRNG is set arbitrarily to stream zero when the Horn's test algorithm begins by the scripted line `rng(0)`, returning the PRNG to its default setting of using the Mersenne twister algorithm; its period is 2^{19937}-1) (Matsumoto, 2011). Resetting the PRNG allows control over duplication of results should similar batches of data be required. Once the script points to `rng(0)` it is not called again until after execution halts and before a new round of MCS begins – it is never reset when the system state is busy.

Readers wishing to adapt this routine to their use should determine what version of MATLAB they are using. If `rng` causes execution halts (errors), legacy random number generator functions will be required (such as `rand` and `randn`). Another consideration: Code that 'flip-flops' between current and legacy random number syntax should reset the random number state using the command `rng default`. For more information, see the MATLAB User Guide, specifically the documents *Updating Your Random Number Generator Syntax* and *Controlling Random Number Generation*.

3.4.2. The Importance of Selecting Sufficient Monte Carlo Iterations

There are two reasons a sufficiently large value of *k* is not only desired, but critical to successful algorithm implementation. The first is there has to be enough opportunity for the means of the eigenvalues $\left(\bar{\lambda}\right)$ to approximate the steady state at θ_i. Steady state is a somewhat of a misnomer because Horn's procedure does not require time series data nor does it experience a warm-up transient period in the simulation. Larger values of *k* move to convergence at the true mean of each eigenvalue, yielding increased precision at the cost of longer processing time. Smaller *k* requires less computation effort and, considering the amount of random data that must be found, is desirable but not so at the expense of inaccurate results for $\bar{\lambda}$. A balance of timeliness and accuracy is required.

Exploratory runs with varying levels of *k* show an exploitable trend. Figure 3.1 illustrates the convergence of the MEVs as *k* is increased. Panel A shows *k* from 10 to 10,000. Note *k* = 10 (magenta line) does not band tightly with the others; it is a poorer fit. Visually, it appears *k* = 100 is a good balance between convergence and computation effort. Panel B shows the chosen *k* = 100 compared to *k* = 10,000; the fit is satisfactory. Because of these reasons (fit vs. effort), *k* = 100 is used in the algorithm found in Figure 3.3 for the remaining thesis work.

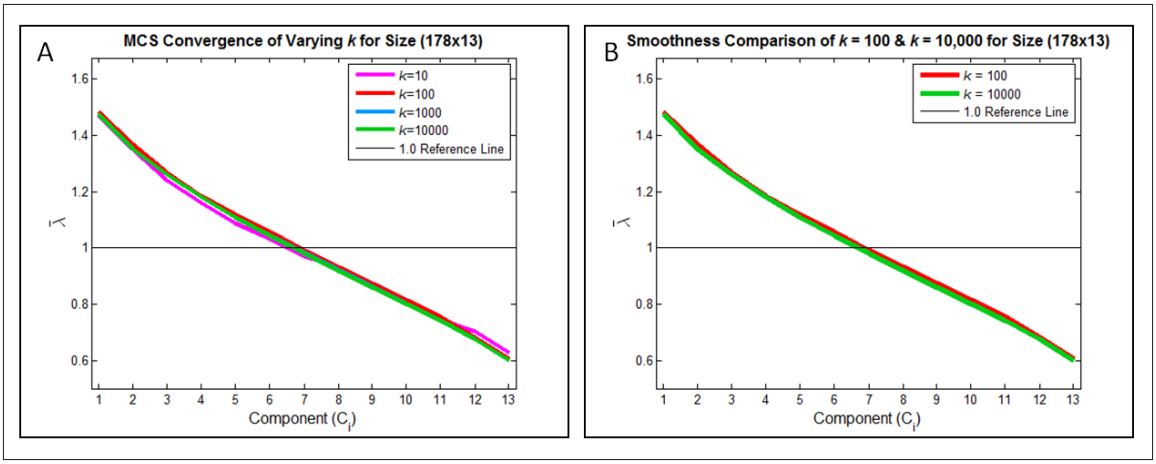


Figure 3.1. Different values for the number of Monte Carlo simulation iterations on a common size of random data.

3.5. Flowchart of Horn's Algorithm for Random Data

It may be of value to view the algorithm from a function perspective; that is, once the input parameters *n*, *p*, and *k* are defined, everything is self-contained to generate in-turn an output for the next step of the solution. The flowchart and pseudo code shown in Figure 3.2 were easily scripted into a MATLAB .m file. The script shown in Figure 3.3 was written as a function because of its specialized purpose. Figure 3.3 is an executable MATLAB .m file – the inputs are values for *p*, *n*, and *k* – for Equations (3.1) - (3.5). The resultant vector `EigenMean` is applied in developing plots, graphs, the lookup table values, second-order model coefficients. It encapsulates the information needed to produce an estimation of Horn's curve.

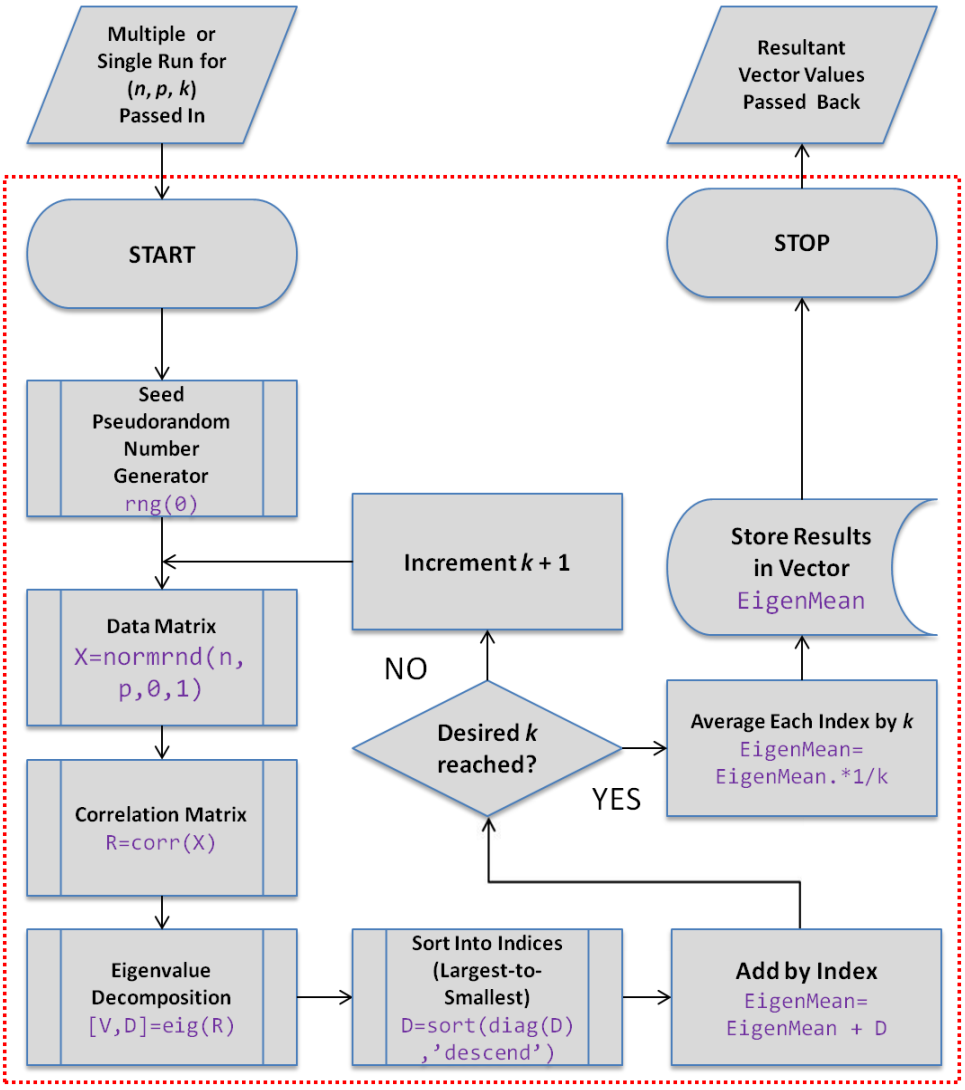


Figure 3.2. Flowchart diagram of the MATLAB algorithm for Horn's test on random data. The red dashed border indicates modular functionality.

```
%******Horn's Procedure using RANDOM DATA in simple MATLAB script.*********
%******Does not contain plotting or error checking routines.***************
%
%Variables:
%p = # of variables; set through interactive input or hard coded
%n = # of observations; set through interactive input or hard coded
%k = # of Monte Carlo simulation iterations
%X = matrix of random variables ~NID(0,1)
%V = eigenvectors returned from 'eig' call.  Used for component loading
%    and confirmatory testing.  Not needed for Horn's curve
%D = eigenvalues of R returned from 'eig' call.  Essential!
%->Note 1: Sampled data is not averaged; it is what it is
%->Note 2: X must lead to invertible R for 'eig' to complete
%          Therefore, X might need to be manually conditioned
%          for multicollinearity, NaN, linear dependence, etc
%initialize variables/starting conditions
EigenMean = diag(zeros(p))';        %preallocate array of mean eigenvalues
rng(0);                             %seed the random number generator
%loop runs Monte Carlo sim (random data draws) on the chosen size data
for i = 1:k
    X = normrnd(0,1,n,p);           %random data matrix X size n x p
                                    %elements of X are iid ~N(0,1)
    R = corr(X);                    %Correlation matrix of X
    [V,D] = eig(R);                 %(V) eigenvectors, (D) eigenvalues
    D = sort(diag(D),'descend')';   %sort 'eig' result from large to small
    EigenMean = EigenMean + D;      %Add each eigenvalue by array index
end
%*********************************Result*********************************
%'EigenMean' is the vector of eigenvalue means over all k iterations.
EigenMean = (EigenMean.*(1/k));
%************************************************************************
%
%end of program.
```

Figure 3.3. Horn's test algorithm for random data in MATLAB script.

3.6. Characteristics of Scree Lines

Now that we have a working algorithm, we are motivated to pause and perform a progress check. Horn provided a figure in his paper (1965:184) to illustrate parallel analysis in action. It is reproduced here in Figure 3.4 with some embellishments to highlight the key features. Curve A (blue circle) is the scree line from actual data, curve R_a (red circle) is from the idealized random data, and the intersection of the curves (green rectangle) indicates estimated component dimensionality. Because we do not have the original data set, only the *n* x *p* size, it is not possible to reproduce curve A. However, it

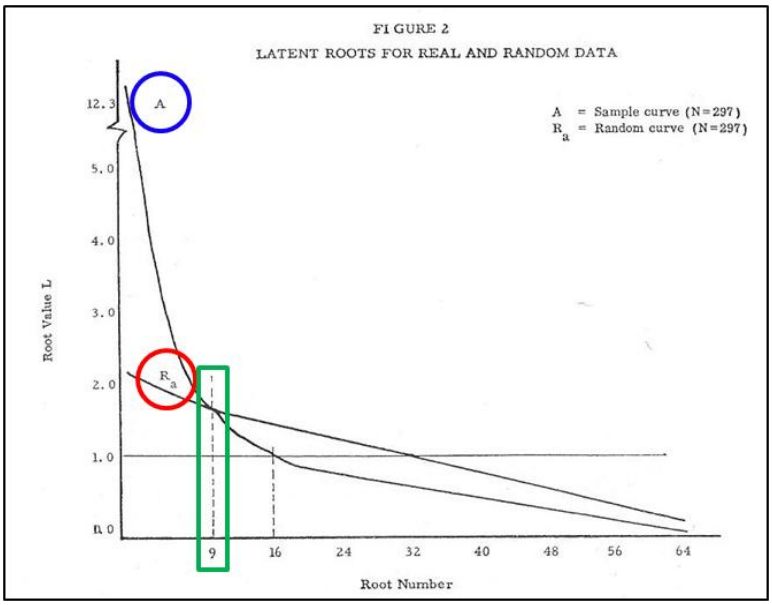


Figure 3.4. Horn's original figure of the theory put into application.

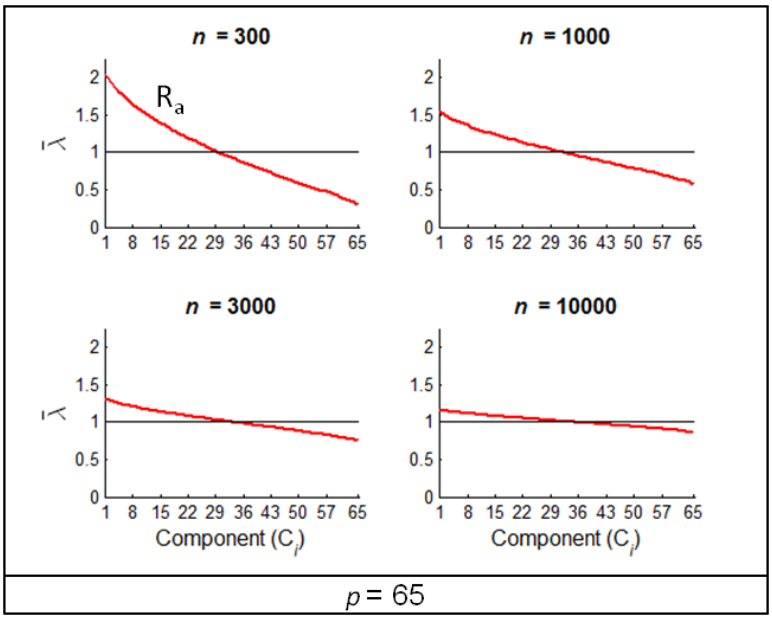


Figure 3.5. Reproduction of Horn's illustration, this time with varying observations n. Notice the convergence of rotation in the slope towards unity.

is possible to produce curve R_a simply by knowing the data size. We do this in the upper left subplot of Figure 3.5. To help visualize the concept that as *n* approaches infinity the slope of the ideal curve goes to zero at unity, we can fix *p* at a particular value and

observe what happens as we increase *n*. We can see here that the differences between the components becomes less and less as observations increase; this is due to an equalizing of total variance among the components. The dominant components 'lose' a percentage of variance, the smaller ones 'gain' variance. In theory, if the sample size were taken to infinity, they will all account for an equal portion of the total variance.

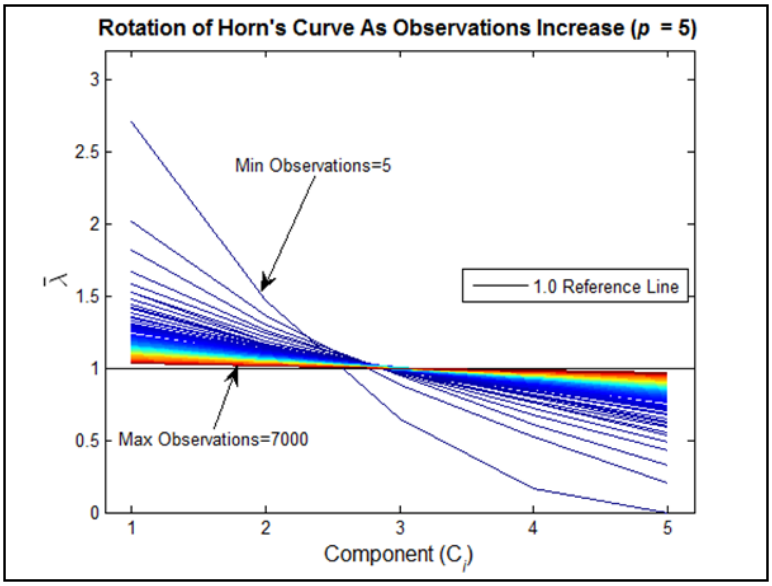


Figure 3.6. Fixed p=5 and varying n through 220 increments from 5 to 7,000.

To further our understanding Figure 3.6 is introduced, illustrating the progression of the slope towards zero as the variable is held fixed. This time, we observe a small number of variables ($p = 5$) and observe the rotation behavior in both speed of convergence (large jumps between ordinate axis values – the mean eigenvalues – indicate rapid movement; no gap shows very little change in slope). Notice that the density of curves is closest to the 1.0 reference line. The ratio of *n* to *p* varies from 1:1 at the start of the sequence to 1400:1 when the 7,000 maximum observations are reached.

If we repeat the process, this time for using Horn's example of $p = 65$, we see

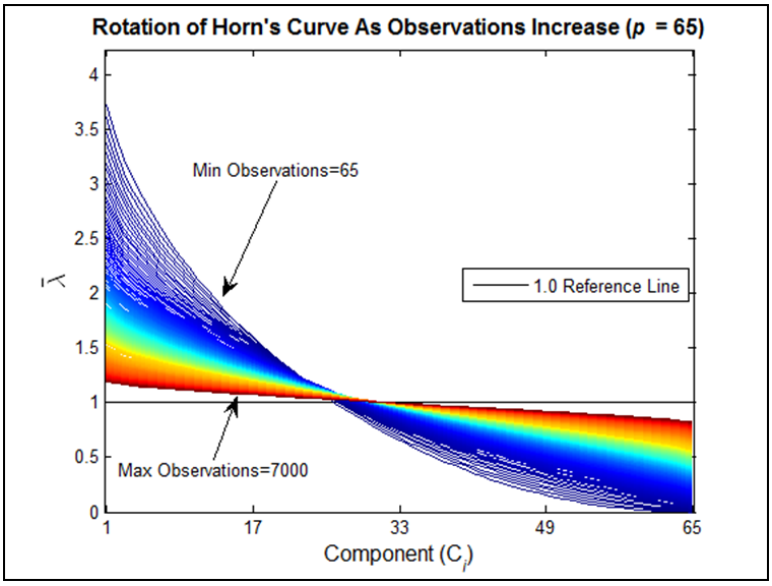


Figure 3.7. Fixed p=65 and varying n through 208 increments from 65 to 7,000.

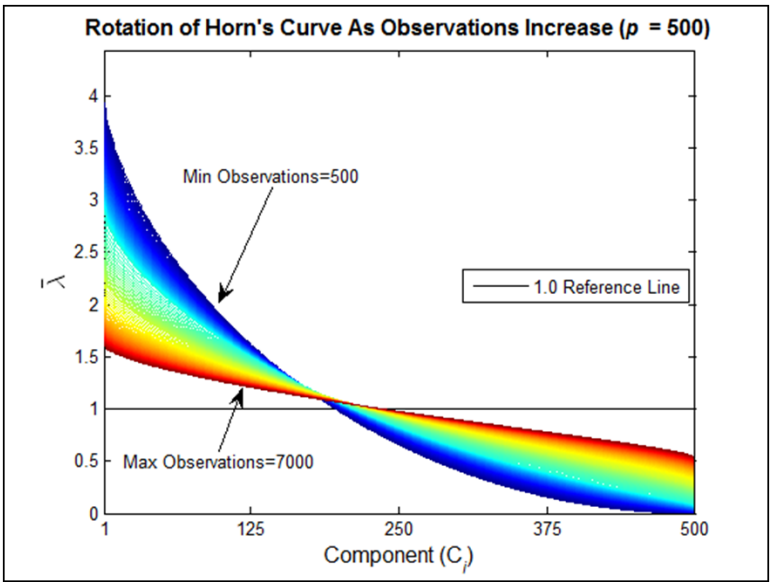


Figure 3.8. Fixed p=500 and varying n through 121 increments from 500 to 7,000.

similar results (Figure 3.7) without large jumps towards zero slope and the convergence towards unity is not as tight. The starting *n:p* ratio is 1:1; at completion it is 108:1.

The final graphic (Figure 3.8) in this group is $p = 500$ variables plotted against the same maximum number of rotations. We observe no unexpected trends.

The jump size is no longer discernible and the rotations are exhausted further away from the reference line. The starting *n:p* ratio is 1:1 and at completion is just 14:1.

3.7. Rationale for Excluding Underdetermined Data

When there are more variables (columns) than observations (rows) in a matrix full rank is not possible. If $p > n$ at least one of the eigenvalues is zero and the determinant of **R** = 0. This case is called *underdetermined*. Figure 3.9 is a comparison of curves for underdetermined ($p > n$), overdetermined (or adequate; $p < n$), and minimum ($n = p$) fitted data.

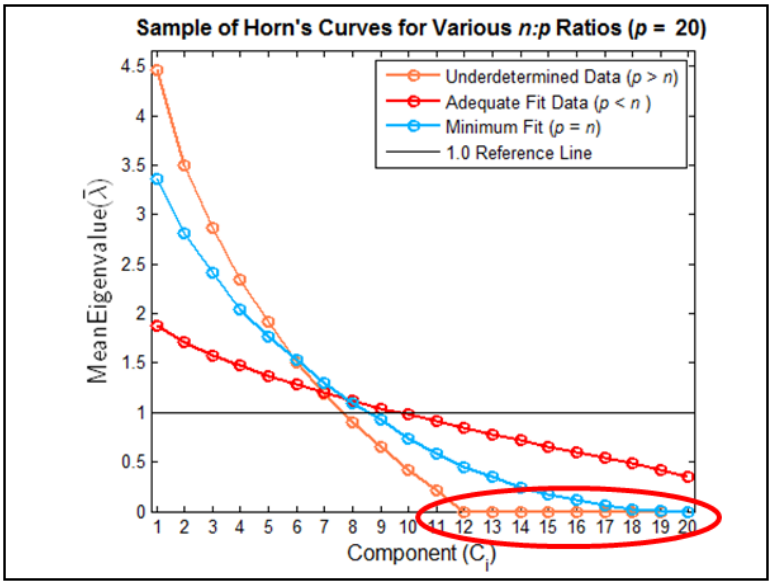


Figure 3.9. Comparison of underdetermined, adequate, and minimum fit random curves. In this example, Components 13-20 have trivial mean eigenvalues (red ellipse) when n=12.

In Figure 3.9 we see an example of each type of these curves. Notice that the last eight MEVs equal zero for the underdetermined data (orange curve, lower right corner). This is not a random event; the initial conditions for the underdetermined data are $p = 20$ and $n = 12$.

There are three reasons to be concerned with inclusion of underdetermined data into the lookup table.

(1) We desire useful components for dimensionality reduction. If we have multiple zero-valued components, information has already been lost before starting PCA.

(2) MEVs do not meet our criterion that the curve be strictly monotonically decreasing $\bar{\lambda}_1 > \bar{\lambda}_2 > ... > \bar{\lambda}_{p-1} > \bar{\lambda}_p$ in the interval $\left[C_1, C_2, ..., C_{p-1}, C_p\right]$ because we find $\exists C_i = 0 \ \forall i : i \in \{p > n\}$. In the sample of curves shown in Figure 3.9, components $C_{13} = C_{14} = ... = C_{19} = C_{20} = 0$ and therefore the requirement is violated.

(3) To capitalize on what is useful in terms of relevant multivariate research, only data of practical value will be pre-processed into the lookup table. We observed in Section 2.6 that $p > n$ sized datasets are uncommon in the published datasets.

Therefore, because of these reasons, $p > n$ sizes are excluded from the analysis.

3.8. Flowchart of Horn's Test for Sampled Data

The algorithm for sampled data is well-represented in a flowchart format. In sampled data, n x p are defined by **X** and random data draws of k are not required. Provided **X** is adequately conditioned for correlation and **R** is invertible, this algorithm will find the eigenvalues. Error checking the input **X** is not shown here; it takes place inside the main program directing input/output and all interactions with the user. The red dashed border signifies the code is modular in design and can be written as a function or

placed inside the main program. Note that while the code is complete and standalone, the intent is for it to work in concert with pre-processed random data (refer to the flowchart for random data in Figure 3.2 and the MATLAB script in Figure 3.3). The two algorithms – random data and sampled data – are called consecutively and they have an equal role in producing a comprehensive solution. While each is algebraically independent, the interweaving of the two is required for a full assessment of how many components to extract for PCA of the problem presented.

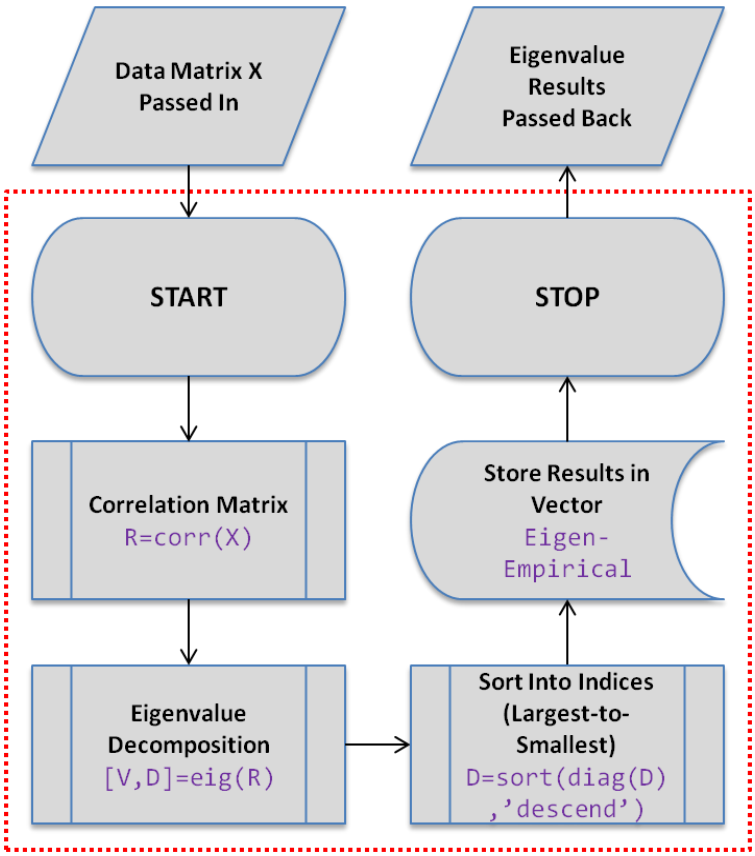


Figure 3.10. Flowchart diagram of MATLAB script for Horn's test on sampled data. The red dashed border indicates modular functionality.

```
%******Horn's Procedure for SAMPLED DATA in simple MATLAB script.**********
%******Does not contain plotting or error checking routines.***************
%
%initialize variables/starting conditions
%X = matrix of study data; observations in rows, variables in columns
%    Must be preloaded in memory or from command "load(filename, '-.mat)"
%    If you use other data storage variables, replace the 'X' in
%    "R = corr(X)" with your data variable
%V = eigenvectors returned from 'eig' call.  Used for component loading
%    and confirmatory testing.  Not needed for Horn's curve.
%D = eigenvalues of R returned from 'eig' call.  Essential!
%->Note 1: Sampled data is not averaged; it is what it is
%->Note 2: X--the data matrix--must lead to invertible R for 'eig' to
%          complete.  Therefore, X might need to be manually conditioned
%          for multicollinearity, NaN, linear dependence, etc.
R = corr(X);                          %Correlation matrix of sampled X
[V,D] = eig(R);                       %(V) eigenvectors, (D) eigenvalues
%*********************************Result*********************************
'sev' is the vector of sample data eigenvalues for all components
sev = sort(diag(D),'descend')';  %sort result D large to small
%************************************************************************
%
%end of program.
```

Figure 3.11. Horn's test algorithm in MATLAB script for sampled data.

The reader should note that Figure 3.11 contains code applicable to *sample data only*. Unlike the flowchart of Figure 3.2 and code in Figure 3.3, there is no looping through MCS draws and averaging the eigenvalues by index. Real-world data will likely need to be conditioned – MATLAB can store and manipulate missing or incomplete matrix entries but such data (NaN and Inf) is indigestible to the correlation operator and toxic to the `eig` function. This code is compact enough to remain in the main script – there is no dedicated function for it.

3.9. Interpolation Lookup Table

In Section 2.6 we examined the need for a suitable region-of-interest (ROI) from which to define the sizes of data we wish to evaluate. We did this to meet practical limitations but also to observe the stated objective of parsimonious implementation. The

most direct approach to a solution is to perform Horn's procedure for every point in the ROI; however, not only is the *n* x *p* space large, MCS of many successive iterations may take hours to complete for just one point (*p*, *n*). We are now motivated to ask "How much of the ROI data do we need to pre-process? Is there a way to achieve satisfactory results without preprocessing the MEVs for *every* inclusive point (*p*, *n*) in the ROI?"

In turns out that with careful selection of the distance of intervals between adjacent coordinates (*p*, *n*) we can compute (or map) a fraction of the points into a database lookup table and then use the method of linear interpolation to instantaneously estimate the unknown points on an as-needed basis (say from a user-supplied input). Constructing a meshed ROI lightens the data density without negatively impacting the accuracy of the solution. (We will see that the lookup table for a gridded ROI is still bulky and took longer than eight days of dedicated processing time to complete.)

The lookup table has several elements that must come together: Lookup table dimensions, granularity of the mapped (*p*, *n*) space, searching of nearest neighbors and creation of surrogate curves, and piecewise linear interpolation of the solution from nearest neighbors. Each of these concepts is explored in detail in the following sections.

3.9.1. Dimensions of Data Matching and Search Configurations

Data matching refers to locating (*p′*, *n′*) in the lookup table; both, one, or neither value may be already mapped (i.e., found in a header column for a row of pre-processed MEVs). How the algorithm handles the search is not complex, but it is exhaustive in that the correct subset of data from the lookup table needs to be isolated prior to interpolation.

The matter of mapping each (*p*, *n*) in the ROI is self-defeating because we end up

with an oversized database at the expense of considerable upfront CPU time – an impractical solution. What is needed is a database of manageable size coupled with a fast search algorithm providing accurate data to the interpolating function that in return, gives a reliable estimation of Horn's curve.

To orient understanding of how the lookup table search routine categorizes a user-selected point of interest (p', n'), there are five cases that are possible for a given (p', n') input. As shown in Table 3.1 they are:

(1) neither p' nor n' are in the table;

(2) p' is in the table, n' is not;

(3) p' is not in the table, n' is;

(4) p' and n' are both in the table; and

(5) Either one is or both (p', n) are out of range of the table.

Table 3.1. Point-of-interest (p', n') input cases and search method sections.

p'	***n'***	***Case Number***	***Section(s)***
No direct match	No direct match	1	3.10.2, 3.10.3
Direct match	No direct match	2	3.10.1, 3.10.2, 3.10.3
No direct match	Direct match	3	3.10.1, 3.10.2, 3.10.3
Direct match	Direct match	4	3.10.1, 3.10.2, 3.10.3
Either or both out of mapped data bounds		5	N/A – Invalid

3.9.2. Lookup Table Granularity

In Section 3.6 some examples of what happens to the slope of Horn's curve as the number of observations n increases for a fixed variable p were demonstrated. To differentiate the progressive decrease in slope as n increases, a color scheme shifting from deep blue to dark red as was chosen. Dark red indicates a Horn's curve with slope

close to zero (that is, they lie close to the horizontal at $\lambda = 1.0$). Deep blue signifies the number of observations *n* is approximately equal to the number of variables *p*.

The takeaway from Figures 3.6 - 3.8 is after some sufficiently large value of *n* the change in slope at (*p, n*) does not differ significantly from that of (*p, n*+1). Perhaps taking *n* at some larger interval (*n*+10, for example) will save processing time, decrease storage density, all with no loss of accurately estimating Horn's curve for (*p, n*).

At this point some exploratory runs for a suitable range of granularity are in order. There are two dimensions (*p, n*) in the data but three possible decisions because the interaction of *p* and *n* need to be considered. In other words, a solution (granularity interval) that works well in one dimension may not work well in the other *and* the power of the (*p, n*) interaction to provide accurate Horn's curve estimation for (*p′, n′*) should be significantly high enough that misleading or inconclusive results are not presented. Fortunately, it turns out that thoughtful selection of granularity in the two dimensions negates concern regarding the combined interaction.

The behavior of Horn's curve as it rotates on $\lambda = 1.0$ about *p*/2 has already been discussed, so we should expect to see it again in the exploratory runs (and we do). What is new is how many observations each variable requires to force the slope of Horn's curve to near zero; apparently, there is a ratio of *n/p* that will give us some idea of 'how much n' for 'how much p' we need if we have a target slope in mind. Recall that the sampling theory behind Horn's procedure is infinite size of *n* and *k* is needed to reach zero slope; that is, all eigenvalues are 1.0 in the population (*n* being the dominant parameter). That statement is not being tested here; the goal is be 'good enough' in

practical application for the curve estimation.

In Figure 3.12, a total of four histograms are used to present individual views of three values of *p* (5, 250, and 500) and the collective set of all *p* variables. The histograms chart the frequency of movement of the first MEV as increasing observations are put into the `EigenMean` algorithm and *p* is held stationary. The rationale behind this analysis is the first eigenvalue is always the largest and it undergoes the greatest change in position as the curve sweeps towards the reference line at 1.0. What the curve shows is, for the selected size of the lookup table, increasing values of *p* 'push back' from 1.0. For instance, when *p* = 5 (upper left subplot in Figure 3.12), approximately 150 of the 220 observation inputs (68%) are nearly equal to one (the large dark red bar). When we look at *p* = 250, we see the minimum value reached is near 1.5 and when *p* = 500, the

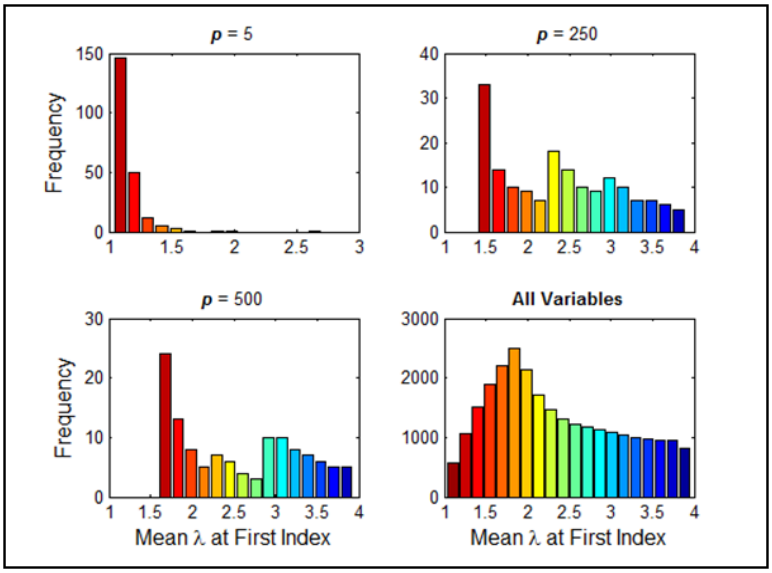


Figure 3.12. Histogram of curve convergence towards 1.0 for various values of p. Dark red indicates curves near $\bar{\lambda} = 1.0$.

minimum value is close to 1.6. For all variables, the mode is near 1.8. The visual analysis from this perspective is in agreement with the individual Horn's curve evaluations; mean eigenvalues in the range of (1, 1.8] are reached using the maximum available observation size of $n = 7000$.

The other dimension (number of variables) is best served with a consistent step size granularity. The number of variables sets the width of the curve; there is no maximum p leading to some change in characteristic of the curve slope. We also know that some number of the eigenvalues will lead to a determination of principal components; therefore, there is more to give up in choosing too high a granularity in the variables than by spending time mapping the MEVs for n x p random data at finer intervals. If the interval for p is too wide, we could lose clarity on the number of principal components because interpolation truncates the number of p components to the lowest nearest neighbor variable found (see $p^{(-)}$ and `nnlp` in Table 3.4). Therefore, it is advantageous to trade processing time for mapping accuracy when it comes to p.

With all things existing and planned considered – desired size of the table, random data pre-processing time, curve convergence to $\lambda = 1.0$ as observations increase, potential search configurations, and limiting width of the horizontal axis – it was determined to fix granularity for p at an increment of five variables. For the observations, the convergence nature of the curve showed some benefit that as the observations increase, we can move from a finer granularity to a coarser one. Also, we observed that many studies take place fairly close to the origin and along the vertical axis. It is beneficial, without loss of resolution, to increment as shown in Table 3.2.

Table 3.2. Granularity intervals in the lookup table.

p	***n***	**Granularity**
5-1000	5-500	5
	510-1000	10
	1050-2000	50
	2100-7000	100

There is a balance to observe in storage size of the lookup table vs. processing power on the (p', n') of interest on the fly. Earlier in this section the choice of iteration step size was discussed for iterations in the MCS. A similar need exists in determining how the large the grid should be in the (p, n) ROI. Recall that, based upon the published 178 data sizes surveyed during the literature review, 138 (76%) were covered in the $1 \leq n \leq 7{,}000$ observations and $1 \leq p \leq 1{,}000$ (respecting the constraint $n = p$).

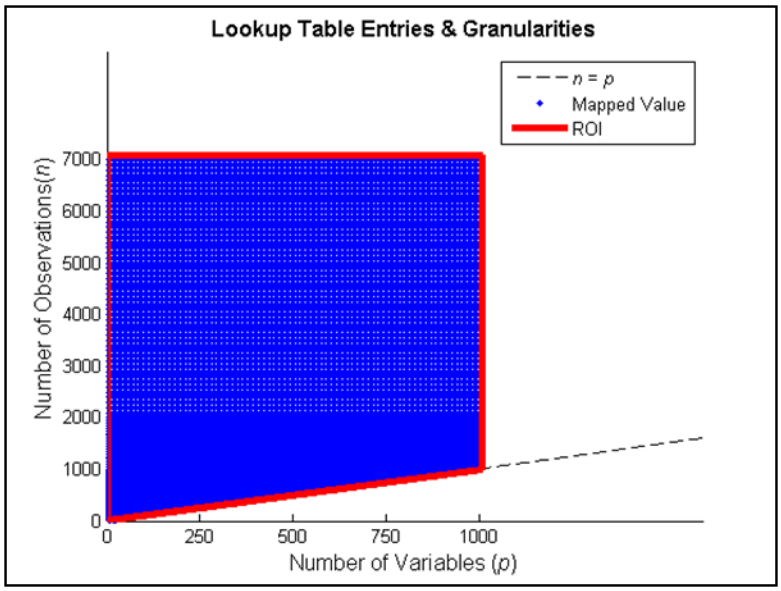


Figure 3.13. Two dimensional representation of the lookup table range. A total of 26,650 rows and 1002 columns (78 megabytes of information) are in the database.

Based upon these factors, the bounded region is ($5 \leq p \leq 1000$) in increments of five variables and ($5 \leq n \leq 7000$) in varying intervals of observations. The ROI is shown in Figure 3.13; changes in color depth correspond to changing granularities. Note that there is too much saturation to distinguish between the granularities of 5 ($5 \leq n \leq 500$) and 10 ($510 \leq n \leq 1000$). Please refer to Figure 2.6 for a wide angle view of the mapped area and the types and density of published studies that 'reside' there.

3.9.3. Lookup Table Format

Populating the lookup table (referring to it as **T** for convenience) with the desired range of data on the stated granularities was a matter of running the random data algorithm in those intervals. At completion, **T** had grown to 26,650 rows and 1,002 columns. The two additional columns are incorporated into **T** for bookkeeping; they identify what (*p*, *n*) coordinate pair a row of MEVs belongs to. In the lookup table, the lowest numbered rows have the highest variables – the variables *p* are sorted in descending order. For the number of observations, the opposite is true: they are sorted in ascending order. To make **T** column equivalent, zeros are added in the rows beyond the number of columns filled by MEV data. The zeros are used as 'filler' because the number of $\bar{\lambda}$ entries equals the number of variables (which change throughout **T**.)

A sample of this structure is given in Table 3.3. In the rows that contain $p = 5$ in the first column (*p*) there are 0's in columns $\bar{\lambda}_6$ through $\bar{\lambda}_{1000}$. The pattern is similar for $p = 500$; columns for $\bar{\lambda}_{501}$ through $\bar{\lambda}_{1000}$ have '0' entries.

Also visible in Table 3.3 is the descending order in the first column (*p*; high-to-low) and ascending order arrangement in the second column (*n*; low-to-high). This

schema was adopted to put the zero cells to the right side and to the bottom of the matrix. Therefore, **T** is largely sparse. Obviously there is a need to search the rows of **T** for the closest match to (p', n'). A nearest neighbors search algorithm completes this task.

Table 3.3. Compressed sample of entries from the lookup table T. Columns extend to $\bar{\lambda}_{1000}$. Diagonal dots indicate sparse columns. Header columns are p and n.

p	***n***	$\bar{\lambda}_1$	$\bar{\lambda}_2$	$\bar{\lambda}_3$	$\bar{\lambda}_4$	$\bar{\lambda}_5$	$\bar{\lambda}_6$	…	$\bar{\lambda}_{995}$	$\bar{\lambda}_{996}$	$\bar{\lambda}_{997}$	$\bar{\lambda}_{998}$	$\bar{\lambda}_{999}$	$\bar{\lambda}_{1000}$
1000	1005	3.906	3.901	3.863	3.829	3.801	3.774	⋯	1.28e-4	9.37e-5	6.56e-5	4.09e-5	2.16e-5	8.71e-6
1000	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
1000	7000	1.891	1.879	1.863	1.855	1.849	1.843	⋯	0.405	0.402	0.400	0.397	0.394	0.390
995	995	3.964	3.992	3.875	3.84	3.809	3.782	⋯	6.89e-18	0	0	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱	⋱	0	0	0	0	0
500	500	3.931	3.852	3.791	3.739	3.694	3.653	⋱	0	0	0	0	0	0
500	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱	0	0	0	0	0	0
500	7000	1.599	1.586	1.576	1.568	1.562	1.555	⋱	0	0	0	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱	⋱	0	0	0	0	0	0
10	10	3.081	2.230	1.680	1.192	0.829	0.535	⋯	0	0	0	0	0	0
5	5	2.707	1.475	0.652	0.167	1.04e-17	0	0	0	0	0	0	0	0
5	⋮	⋮	⋮	⋮	⋮	⋮	0	0	0	0	0	0	0	0
5	7000	1.034	1.016	1.000	0.984	0.966	0	0	0	0	0	0	0	0

3.9.4. Datasets Having Small Number of Variables ($2 \leq p \leq 4$)

PCA of micro datasets does occur. For instance, Sir Ronald Fisher's 'Iris' dataset (Frank & Asuncion, 2010) consists of four variables. For p' in the range of $2 \leq p \leq 4$, the `HornsMethodSampledMEV.m` and `HornsMethodSampled2OM.m` algorithms bypass the NN searches and directly calculate the (p', n') eigenvalues. Because these datasets have few variables, computed results using a direct application of Horn's algorithm are received with little delay. Note that *exploratory runs* using this range of p are not possible with the algorithms presented because `HornsMethodRandomMEV.m` and `HornsMethodRandom2OM.m` each depend upon their respective lookup tables and $p < 5$ will be rejected at the input menu. The `EigenMean.m` function will evaluate

p of this size; however, it does not graph the result. The solution is a future work topic.

3.10. Nearest Neighbors Search Algorithm

Because of the intervals between lookup table entries, we use a two part procedure called nearest neighbors (NNs) search to capture the position of (p', n') as it relates to known values in T. Once the known values have been found, they are passed to the interpolating function to be 'read between the lines,' thereby estimating the MEVs defining Horn's curve for the (p', n') coordinate pair. This section concentrates on the NN search and interpolation is discussed in Section 3.11.

By nearest neighbors we are referring to the first mapped value in **T** greater than and less than that of each p' and n'. Table 3.4 lists the NN search variables and the roles they have in the algorithm. For continuity with the MATLAB code in Appendix II, the MATLAB variables are also provided in Table 3.4. The mechanics of the algorithm are straightforward. The goal is to 'sandwich' (p', n') such that $p^{(-)} \le p' \le p^{(+)}$ and

Table 3.4. Nearest neighbor search variables naming schema.

Variable	Role
$p^{(-)}$	Lower nearest neighbor variable. In MATLAB it is `nnlp` for "`n`earest `n`eighbor `l`ower `p`"
$p^{(+)}$	Upper nearest neighbor variable. In MATLAB it is `nnup` for "`n`earest `n`eighbor `u`pper `p`"
$n^{(-)}$	Lower nearest neighbor observation. In MATLAB it is `nnln` for "`n`earest `n`eighbor `l`ower `n`"
$n^{(+)}$	Upper nearest neighbor observation. In MATLAB it is `nnun` for "`n`earest `n`eighbor `u`pper `n`"
Row Identifiers (all)	During scan of **T** for a nearest neighbor, row values are returned. Adding '`r`' to the upper/lower NN variable names provides row information on where the NN variables are located. For example, the NN search of **T** provides `rnnlp`, `rnnup`. The NN search of **S** yields `rnnln`, `rnnun`. (Section 3.10.3)

$n^{(-)} \le n' \le n^{(+)}$. Searching is done in two parts, first in the p column (Column 1) and then proceeding to the n column (Column 2) of **T**.

3.10.1. Boundary Conditions Not Along The Diagonal

Before beginning a search of **T** for $p^{(-)}$ and $p^{(+)}$, the algorithm checks to see if p' is at a boundary by comparing it to the minimum and maximum values of p in the lookup table. The minimum and maximum values of the lookup table are dynamically assigned each time the program starts; therefore, if the boundary values of the lookup table change, the min/max values are updated. If p is found along boundaries, the algorithm makes assignments to the NN variables $p^{(-)}$ and $p^{(+)}$ as given in Table 3.5. Note that this process repeats exactly for n' after $p^{(-)}$ and $p^{(+)}$ have been given assignments.

Table 3.5. Boundary conditions and how to address them in nearest neighbor assignments. ±5 and ±100 are the maximum granularities for p and n, resp.

Condition	Solution
p' = minimum p	$p^{(-)} = \min(p)$ $p^{(+)} = \min(p) + 5$
p' = maximum p	$p^{(+)} = \max(p)$ $p^{(-)} = \max(p) - 5$
n' = minimum n	$n^{(-)} = \min(n)$ $n^{(+)} = \min(n) + 100$
n' = maximum n	$n^{(-)} = \max(n)$ $n^{(+)} = \max(n) - 100$
Either $n^{(-)} = p^{(-)}$ or $n^{(+)} = p^{(+)}$	$n^{(-)} = n^{(+)}$

3.10.2. Boundary Conditions Along The Diagonal

In a moment we will discuss how the algorithm searches first for $p^{(-)} \le p' \le p^{(+)}$ and then $n^{(-)} \le n' \le n^{(+)}$. In Section 3.7 the case was made to not process any $p > n$ sized data. Because all combinations of pairs for $p^{(-)}$, $p^{(+)}$, $n^{(-)}$, and $n^{(+)}$ are needed for

interpolation, the algorithm must be able to detect if an NN assignment is made that violates the constraint. In Figure 3.14, we see a notional out-of-bounds configuration. Because (p', n') was somewhat near the diagonal but not over it, the NN values pulled from **T** are valid at their original positions within the table but creation of new NN coordinate pairs resulted in an invalid combination. For example, in Panel A the coordinate ($p^{(-)}$, $n^{(-)}$) breaks the constraint. The solution is to use what is already known about the NNs and reassign the coordinate as ($p^{(-)}$, $n^{(+)}$).

The reader might notice that we now have <u>two</u> pairs at the same coordinate. It might seem that something has been lost but this is not the case. The linear interpolation method we are using (Section 3.11) is robust in making computations for overlapping lines and lines that cross. Part I of the interpolation method is to construct <u>two surrogate</u>

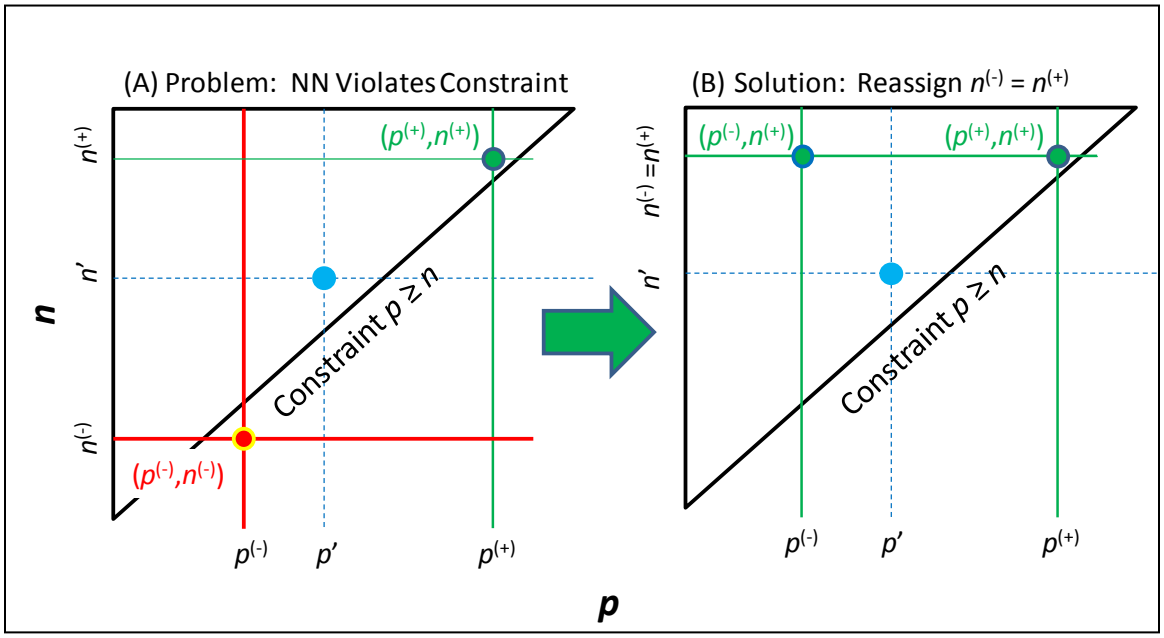


Figure 3.14. Case of out-of-bounds nearest neighbor find. In Panel A, ($p^{(-)}$, $n^{(-)}$) violates the minimum constraint $n \geq p$. Panel B shows the solution is to set $n^{(-)} = n^{(+)}$.

curves and use subsequent interpolation of the surrogate curves to reach the desired solution for (p', n'). In the example shown in Figure 3.14, the interpolation routine will return a surrogate curve that matches the MEVs mapped for ($p^{(-)}$, $n^{(-)}$). Therefore, we do not have to do anything odd or complex to find and fix the constraint violations.

3.10.3. Nearest Neighbors Not At The Boundaries

Returning to our goal of finding $p^{(-)} \le p' \le p^{(+)}$ and $n^{(-)} \le n' \le n^{(+)}$, we search in two parts, first in the variables column and then moving to the observations column. The mechanics of the algorithm are simple: Centering on p', look to find p' in **T**. If there is a direct match, the algorithm records the rows where p' is found, assigns $p^{(-)} = p^{(+)} = p'$ and move out of the search loop for p'. Figure 3.15 illustrates the general process.

If no direct match is found, start a loop counter at 1 and increment p' to find $p^{(+)}$ and decrement p' to locate $p^{(-)}$. We set a loop limit equal to the p granularity (5, for all cases) so that the algorithm avoids entering an infinite loop searching for a value that will never be found (this should only happen if **T** is somehow corrupted). We continue to search **T** iteratively *above and below* p' until mapped values for $p^{(-)}$ and $p^{(+)}$ are assigned. Once $p^{(-)}$ and $p^{(+)}$ are known, rows of **T** are extracted to form **S**, a matrix containing the bookkeeping columns and the MEV data for $p^{(-)}$ and $p^{(+)}$. The reason for creating this smaller matrix is we already have half the information needed for all coordinate pairs of $p^{(-)}$, $p^{(+)}$, $n^{(-)}$, and $n^{(+)}$. There is no advantage to searching for $n^{(-)} \le n' \le n^{(+)}$ in **T**. Because of the sparsity of the lookup table, the rows of **S** are truncated to the number of non-zero columns for $p^{(-)}$. This ensures the rows **S** are free of the 'filler' data used to support the structure of the lookup table – zeros have no practical meaning to subsequent

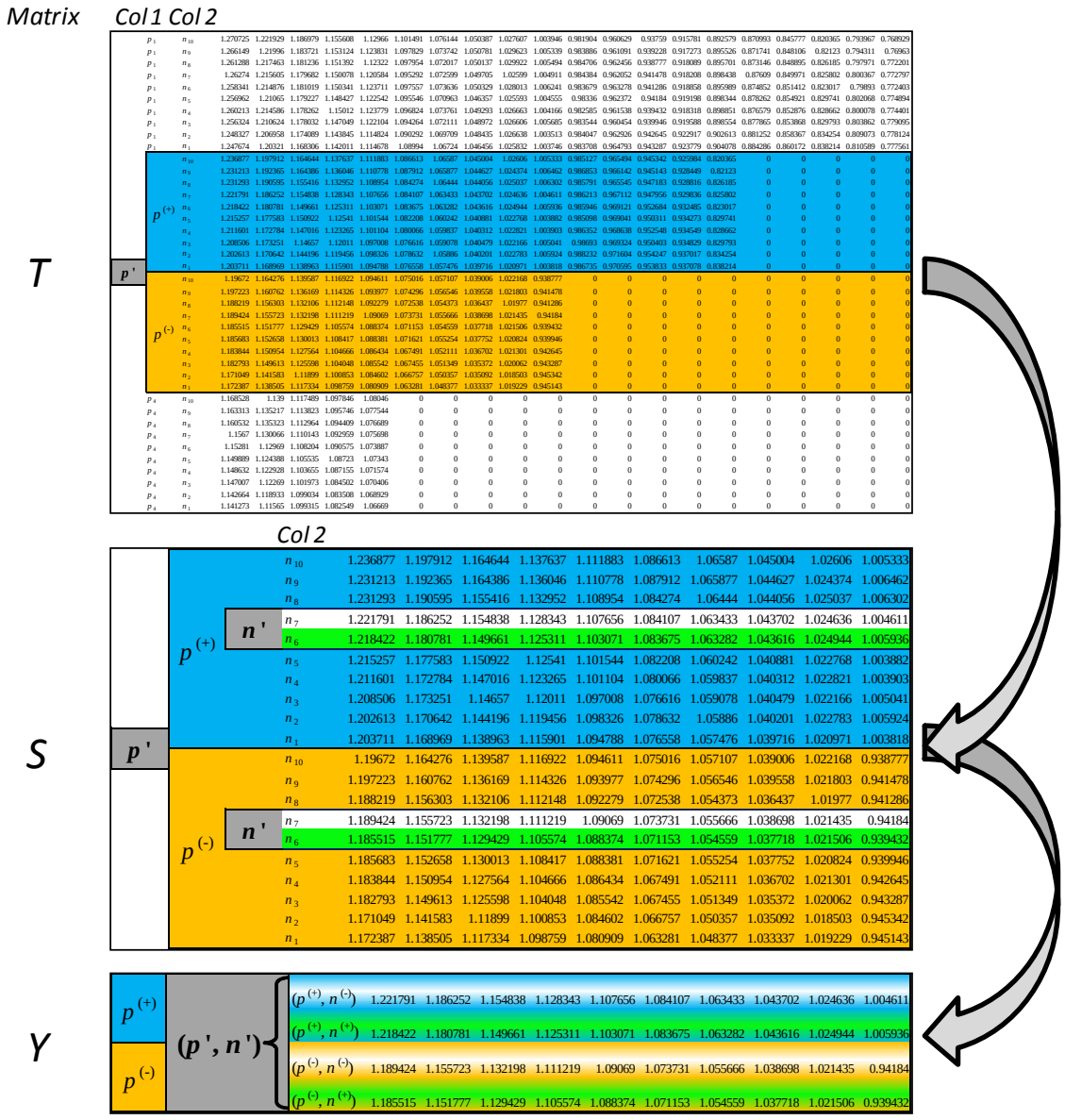


*Figure 3.15. Trimming of the lookup table **T** to sub-matrix **S**, and finally a matrix of only nearest neighbors data, matrix **Y**. Only numeric entries comprise actual **T**, **S**, and **Y**.*

calculations. Column truncation of **S** yields MEVs that extend only to $C_{p^{(-)}}$. However, in Sections 2.2.4 and 3.9.2 we learned the first *p*/2 components is where Horn's curve provides an estimate of random noise in the sample. Therefore, truncation near the last component (and well below $\bar{\lambda} = 1.0$) is inconsequential to our test.

For the input parameter n', the process is similar for that of finding p' in that Column 2 of **S** is searched up and down from n' until $n^{(-)}$ and $n^{(+)}$ are located. If n' is a direct match to a value for n, then $n^{(-)} = n^{(+)} = n'$. The search step size increment is 1 but because of the larger granularity in n the search extends ±100 units away from n'.
At the conclusion of the search for $n^{(-)}$ and $n^{(+)}$, we can reduce **S** to four rows. Each row defines a coordinate pair: $(p^{(+)}, n^{(-)})$, $(p^{(+)}, n^{(+)})$, $(p^{(-)}, n^{(-)})$, and $(p^{(-)}, n^{(+)})$. These four rows are stored in a new matrix, **Y**.

Returning our attention to Figure 3.15, the first column of **T** is searched for $p^{(-)}$ and $p^{(+)}$ and when found, the rows are identified (blue and gold, respectively) and removed from **T** to form **S**. Next, the second column of **S** is searched for the rows containing $n^{(-)}$ and $n^{(+)}$ (green and white rows, respectively) and when located, are removed to form **Y**. The blending of colors in **Y** indicates the combination searches provides the needed information to carry **Y** forward for interpolation.

3.11. Interpolation–Looking Between the Points

Interpolation is the estimation of an unknown intermediate data value by fitting a function through known data values. Given the familiar form $y = f(x)$, the process works backwards to find an unknown function f that represents the known dependent y values from the known independent x values. There are many types of interpolation; the one we are specifically interested in is *piecewise linear interpolation*; piecewise because we are seeking intermediate values at multiple discrete points along a path and linear because we are not fitting any curvature components between those points. The technique is akin to ‘connecting the dots’ which is suitable in this case, as we have

already visualized Horn's curve passes originates at $\left(C_1, \bar{\lambda}_1\right)$ and passes through $\left(C_2, \bar{\lambda}_2\right)$ and so on until terminating at $\left(C_p, \bar{\lambda}_p\right)$. The reader should recognize that averaging the distance between the points is not an acceptable method because the relationship of (*p′*, *n′*) to the nearest neighbors is not necessarily along the midpoint of the data points stored in **Y**. The expression for the piecewise linear interpolation function is given by Quateroni & Saleri (2003)

$$f(x) = f(x_i) + \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}\left(x - x_i\right) \text{ for } x \in I_i \tag{3.11}$$

where $f(x)$ is a function denoted by the interval I_i as $[x_i, x_{i+1}]$ having a set of nodes $x_0 < x_1 < \cdots < x_{p-1} < x_p$. We can consider the nodes as the number of components in our model, each represented by an eigenvalue. The eigenvalues have been sorted, so the inequality requirement holds. We note that this expression meets our need of proximity for the interpolated solution to (*p′*, *n′*) because the quotient weighs the result by differences in both the dependent and independent values between adjacent points. Lastly, the intervals in our problem are finite and closed on adjacent points.

MATLAB has several interpolation functions in its library and the one of interest to the problem at hand is `interp1`, a linear piecewise function of the form

```
yi = interp1(x,Y,xi)
```

where `yi` is the unknown ordinate at abscissa `x` and `Y` are the values of the underlying function at the points of the vector or array of `xi`. Our query is not structured this way; thus far we have treated C_i as the abscissa and $\bar{\lambda}_i$ as the ordinate.

Instead, we are interested in a scree line which is defined by the coordinate pair $\left(C_i, \bar{\lambda}_i\right) \forall i \in \{1, 2, ..., p-1, p\}$ but are providing a (p', n') pair which may or may not be in the lookup table. Although (p, n) is used as input to Horn's algorithm during the random data preprocessing step, determining the (p', n') pair from a scree line is not so clear: We are interpolating one value on the vertical axis for two points along the horizontal axis and `interp1`expects a unique x for every $f(x)$. A multistep approach, one where an intermediate calculation provides a path to a final solution, is required. Therefore, if we can format the input we send to `interp1`, the function will provide the intended result.

3.11.1. Surrogate Curves

In Figure 3.16 there are two places on the graph where the intermediate solution is needed. The first is at the coordinates above (p', n') given by ($p^{(+)}$, $n^{(-)}$) and ($p^{(+)}$, $n^{(+)}$). The second set of coordinates is below (p', n') at ($p^{(-)}$, $n^{(-)}$) and ($p^{(-)}$, $n^{(+)}$). The multi-dimensional interpolation is done this way to maintain order in the nearest neighbor matches. Otherwise, if only one curve is interpolated from all the nearest neighbors, we will have inconsistent results from mixed coordinates. Interpolating two surrogate curves `evu` and `evl` to find the third and final curve maintains the data pedigree.

```
evu(i) = interp1([nnln;nnun],[mevnnln(i); mevnnun(i)],n)
evl(i) = interp1([nnln;nnun],[mevnnln(i); mevnnun(i)],n)
```

Here, `ev` is the unknown eigenvalue along the (`u`)pper or (`l`)ower surrogate curves at component index `i`; `nnln` and `nnun` are the lower and upper NNs for observations; and `mevnnln` and `mevnnun` are the mean eigenvalues for `nnln` and `nnun` at index `i`, respectively. (In Appendix II the `mevnnln` and `mevnnun` are given in terms of rows

of **Y**. These names are used here for simplicity in discussion.) For reference to the nearest neighbor naming schema, please refer to Table 3.4.

Lastly, `n` is *n′*, the unknown we wish to find. MATLAB will perform index operations of vectors without counters; however, we have a special case of a single point on the curve due to multiple responses for one predictor. Iteration takes place for all C_i, `i` = 1 to $p^{(-)}$.

3.11.2. Interpolation of Horn's Curve

When we have both surrogate curves, we can next interpolate the overall solution using `evu` and `evl` as inputs into `getev` as

```
getev(i) = interp1([nnup;nnlp],[evu(i); evl(i)],p)
```

Here, `getev(i)` are the MEVs for (*p′*, *n′*) for each component C_i (still observing `i` = 1 to $p^{(-)}$); `nnup` and `nnlp` are the lower and upper NNs for variables (Table 3.4); and `p` is *p′*, the unknown part of our point-of-interest (*p′*, *n′*). We similarly have to iterate across each point in `evu` and `evl` but when finished have arrived at the desired solution: a completed Horn's curve for the user-supplied (*p′*, *n′*) input.

It is helpful to demonstrate an example of how the interpolation sequence occurs. Returning to Figure 3.16, we see a visual representation of fictitious data from **Y** in the form of four plotted upper and lower NN curves. The two dark blue lines represent upper NN pairs ($p^{(+)}$, $n^{(-)}$) and ($p^{(+)}$, $n^{(+)}$) and the gold lines fix lower NN pairs ($p^{(-)}$, $n^{(-)}$), and ($p^{(-)}$, $n^{(+)}$). Round markers signify the corresponding MEVs for each component C_1 - C_5 with respect to the NN curves. Brackets indicate the range of the anticipated Horn's curve solution for (*p′*, *n′*). No interpolation has yet occurred.

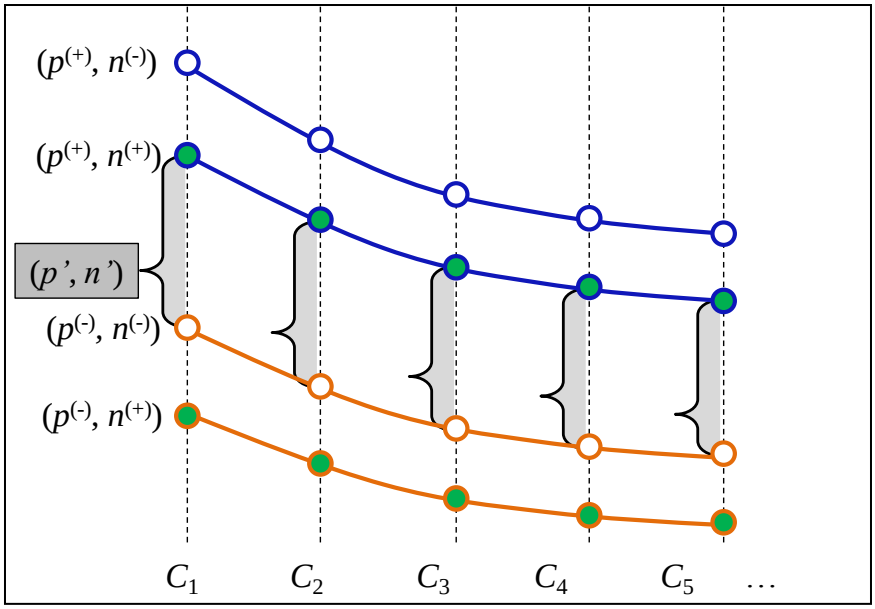


Figure 3.16. Pictorial representation of the upper and lower nearest neighbors curves. Mean eigenvalue data (not shown) are along the vertical axis. Components (C_i) are along the horizontal axis.

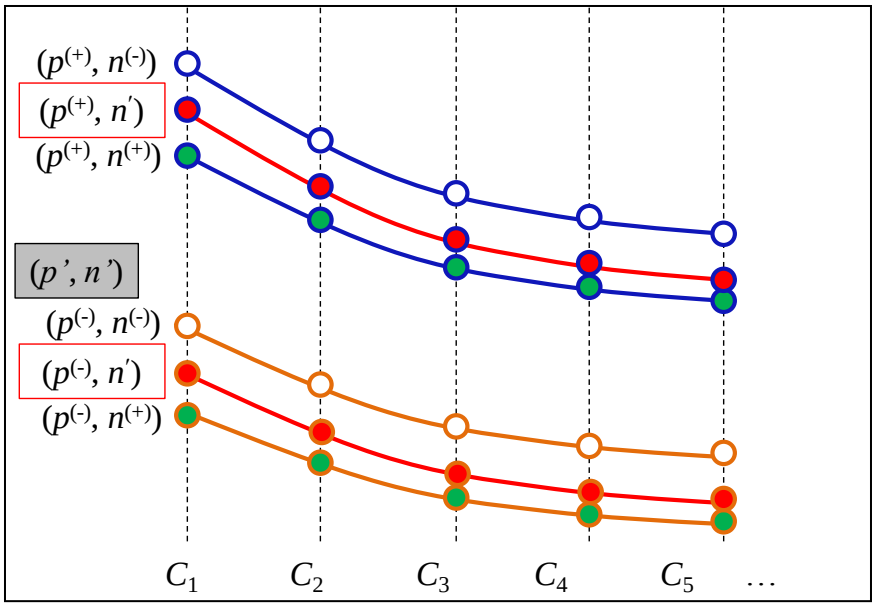


Figure 3.17. Interpolation of the upper surrogate curve at $(p^{(+)}, n')$ and the lower surrogate curve at $(p^{(-)}, n')$. Features created during this step are shown in red.

In Figure 3.17, the upper and lower surrogate curves defining `evu` and `evl` have been found by using two pairs of known curves (the upper and lower NNs from **Y**) for interpolation of each unknown curve at the coordinate ($p^{(+)}$, n') and at ($p^{(-)}$, n'). Features that have changed or been added to on the chart are indicated by red text boxes, lines, and markers. We have located n' from determining $n^{(-)} \le n' \le n^{(+)}$.

The process is repeated in Figure 3.18, this time evaluating the two surrogate curves for the unknown Horn's curve. The interpolation routine `getev` has found p' by determining its relationship as $p^{(-)} \le p' \le p^{(+)}$. We now know each $\bar{\lambda}_i$ for each component C_i, thereby defining the estimate of Horn's curve for (p', n') and can provide the graph for visual analysis. Figures 3.19 - 3.22 show full results rendered in MATLAB.

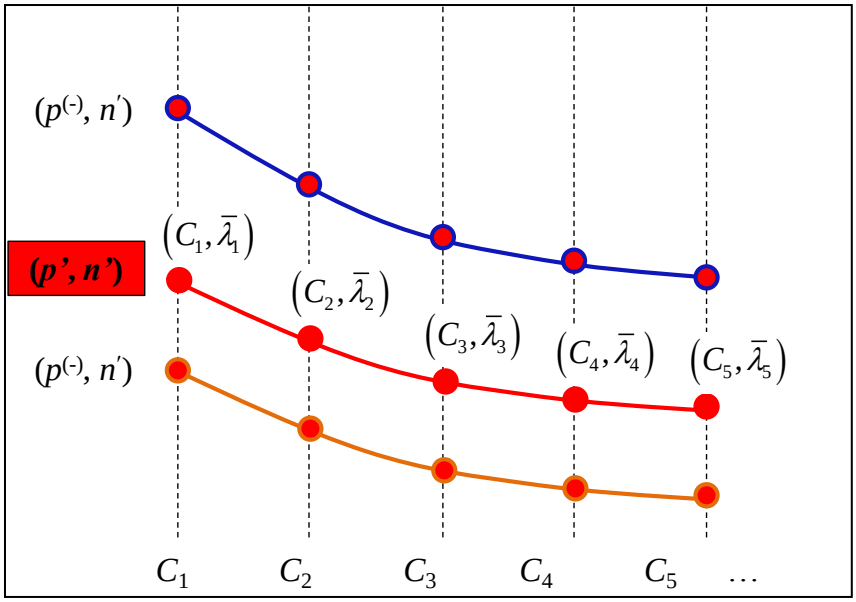


*Figure 3.18. A finished, interpolated solution of the estimated Horn's curve for (p', n'). The solution is shown in solid red; the surrogate curves are in view to orient the interpolation. Each $(C_i, \bar{\lambda}_i)$ is representative of a point along the curve. All $\bar{\lambda}_i$ shown are progeny of the surrogate curves from **Y** and the nearest neighbors extracted from **T**.*

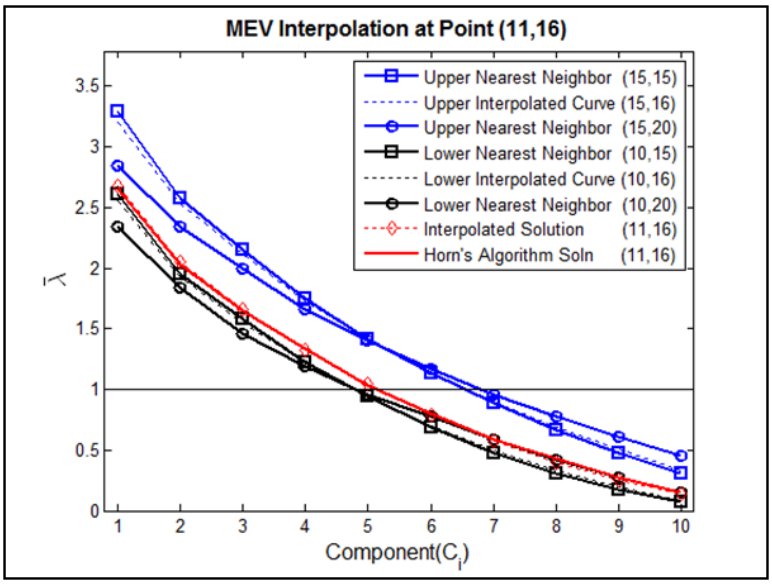


Figure 3.19. A very small dataset. The upper NN curves (blue) cross at p = 5 yet the surrogate curve stays well-banded. This indicates the interpolation routine is robust with regard to which line is above or below the other. The figure legend describes in detail the coordinate pair of each curve drawn.

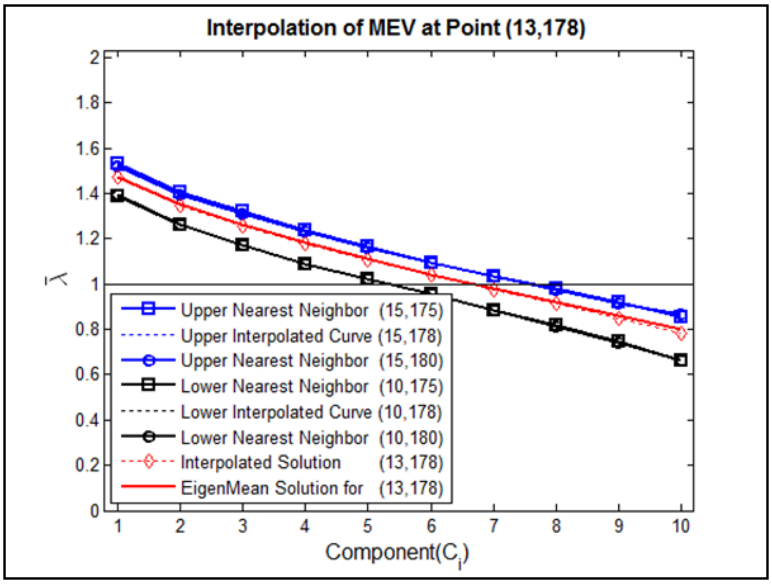


Figure 3.20. A small dataset. Notice the close approximation among the curves.

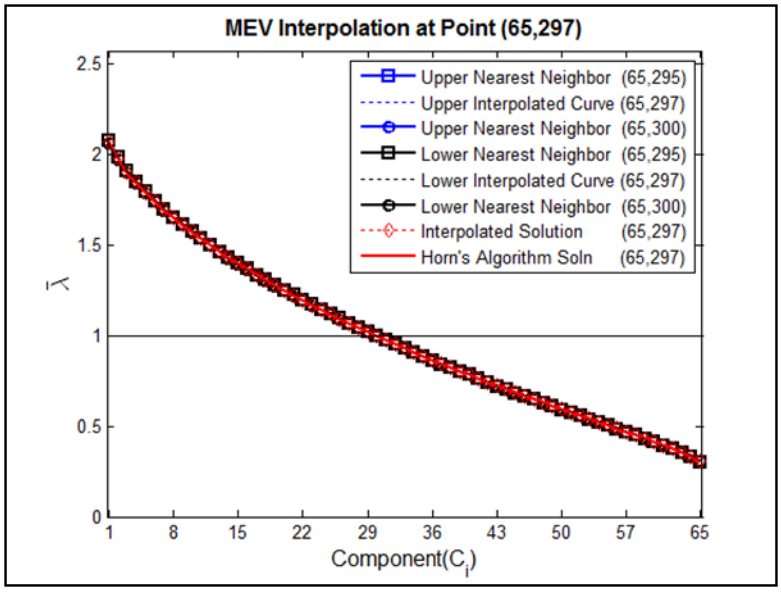


Figure 3.21. A medium dataset. All the curves have converged around the Horn's algorithm solution for random data (solid red line). This graphic uses the same size of data Horn presented in his 1965 paper.

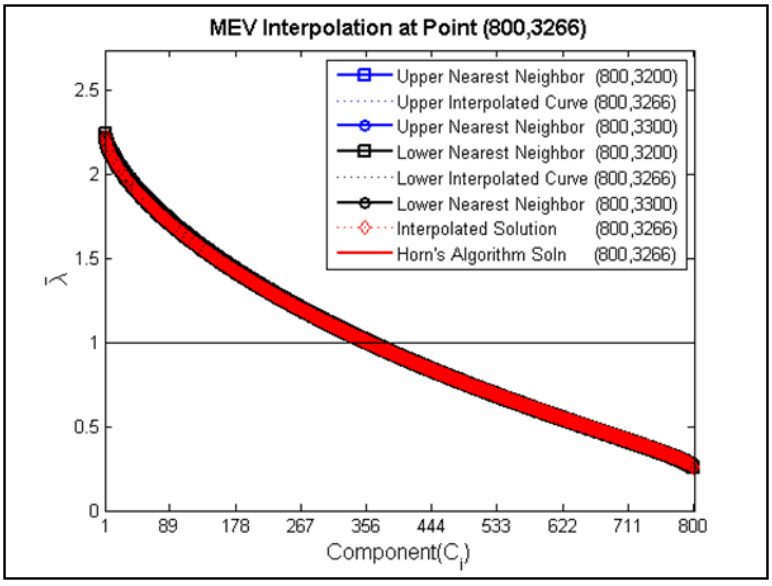


Figure 3.22. A large dataset. There is much less to see in differences between mapped and interpolated in dimensions of this size.

Note that the figures show the upper and lower curves, both from mapped data and interpolated surrogate points. The solid red line running through the middle of the figures represents the interpolated solution and is a direct computation of (p', n') using `EigenMean` (Figure 3.3). Notice that some of the upper/lower nearest neighbor curves cross (Figure 3.19 near $p^{(-)} = 5$). The observation that this does not affect the accuracy of the interpolating function is reassuring. Exploratory results are in agreement with expectations from earlier visual analysis. Note that in this section all interpolation was carried out on random data – no sampled data were used in the analysis.

3.12. Linear Regression Second-Order Model

The primary motivation behind developing a linear regression second-order model (2OM) is to save space. Earlier we learned that the lookup table is sparse – it is full of zeros because the rows only contain data equal to two plus the value of the variable p. If the data already collected (the lookup table) is used to fit a quadratic polynomial for each (p, n) row, then the size of the lookup table can be greatly reduced.

3.12.1. Suitability of A Second-Order Model

The reader might question "Why a second-order model – why not fit a higher-order polynomial?" The answer is in the shape of Horn's curve: its simple characteristics – a slightly bowed line without inflection points – does not require a complex polynomial. Simplicity and parsimony in the model is desired. Three coefficients provides a suitable representation of the curve. More coefficients add data back into the lookup table and does not provide a more accurate solution.

3.12.2. Least-Squares Estimation of Regression Coefficients

The method of least-squares seeks to fit a line through regressor data points **X** by minimizing the differences between the observed responses at **y** and a model predicted response of $\hat{\mathbf{y}}$ (in this case, $\hat{\boldsymbol{\lambda}} = \hat{\mathbf{y}}$). In matrix notation, the model is

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon} \tag{3.12}$$

where $\boldsymbol{\beta}$ are the regressor coefficients, $\boldsymbol{\varepsilon}$ is the error term (**y** and **X** are already defined).

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_p \end{bmatrix}, \ \mathbf{X} = \begin{bmatrix} 1 & x_{11} & x_{12} \\ 1 & x_{21} & x_{22} \\ \vdots & \vdots & \vdots \\ 1 & x_{p1} & x_{p2} \end{bmatrix}, \ \boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_{11} \end{bmatrix}, \ \boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_p \end{bmatrix} \tag{3.13}$$

The form of quadratic model that fits the problem at hand, interpolating an estimated Horn's curve for (*p′*, *n′*), is given by

$$\hat{\boldsymbol{\lambda}} = \hat{\beta}_0 + \hat{\beta}_1\mathbf{C} + \hat{\beta}_{11}\mathbf{C}^2 \tag{3.14}$$

where $\hat{\boldsymbol{\lambda}}$ is a vector of estimated mean eigenvalues and **C** is a vector of components numbered from 1 to *p*. For amplifying information regarding the derivation of Equations (3.12) - (3.14), please see the text by Montgomery, Peck, and Vining (2006) or similar source on linear regression techniques.

The MATLAB built-in function library provides a function called `polyfit` that evaluates vector **C** to return a vector of least-squares estimated coefficients $\hat{\boldsymbol{\beta}}$ at the specified order of the polynomial (2 for a quadratic). Readers may be interested to learn that instead of formatting coefficients in ascending power order, `polyfit` outputs coefficients in a descending power format (β_{11}, β_1, β_0) (Recktenwald, 2000).

We use `polyfit` to evaluate each row of MEVs, thereby reducing the lookup table to only five columns (two for bookkeeping of the coordinate pair and three for the polynomial constant β_0, linear β_1 coefficient, and quadratic β_2 coefficient). The reduction in file size is significant, from almost 80 megabytes of MEV data to just 606 kilobytes of 2OM data. A snapshot of the table is given below. For reference, compare to the original lookup table in Table 3.3.

Table 3.6. Sample of the coefficients lookup table. Total width is five columns–two for coordinate pair bookkeeping and three for coefficients entries.

p	***n***	$\hat{\beta}_{11}$	$\hat{\beta}_1$	$\hat{\beta}_0$
1000	1005	4.5335e-6	-0.0078	3.3774
1000	⋮	⋮	⋮	⋮
1000	7000	6.3377e-7	-0.0019	1.7546
⋮	⋮	⋮	⋮	⋮
500	500	1.8203e-5	-0.0156	3.3876
500	⋮	⋮	⋮	⋮
500	7000	1.2644e-6	-0.0025	1.5139
⋮	⋮	⋮	⋮	⋮
5	5	0.1763	-1.7302	4.2507
5	⋮	⋮	⋮	⋮
5	7000	4.9733e-5	-0.0170	1.0504

To display the data, MATLAB's function `polyval` uses the $\hat{\boldsymbol{\beta}}$ coefficients from a row in the coefficients lookup table to estimate $\hat{\boldsymbol{\lambda}}$ for a particular (p, n) and returns an array we can easily plot. Producing Horn's curve is a simple matter of rendering the ($\mathbf{C}$, $\hat{\boldsymbol{\lambda}}$) coordinates in a figure. We shall see that the 2OM curve is not truncated at $p^{(-)}$ during interpolation as it is for the MEV approach.

3.12.3. Sufficient *k* for Linear Regression

In building the 2OM, the anticipated approach is to use least-squares estimation to find the model coefficients. We need to 'trust' that the lookup table data will adequately

define the regressed line. Therefore, due diligence is required to verify the lookup table entries can be used as a starting point for the model. Note that this is a progress check to verify data is properly conditioned; it is not an exercise to reevaluate if changing the iterations of *k* in the MCS will give different results. (From the discussion in Section 3.4 we already know *k* affects the smoothness of the data.)

In this small scale experiment, the MCS are re-accomplished for varying *k* in powers of 10 from 0 to 3 (10, 100, 1,000, and 10,000). At the end of each *k* runs, the indexed eigenvalues are averaged, stored, and the process repeated until *k* completes the last of the 10,000 iterations. This is done for only one example problem, that of 297x65 (which is, if one refers to Figure 3.4, Horn's sample size from his 1965 paper). Our interest is with how well *k* = 100 'behaves' because it is the size of the MCS iteration

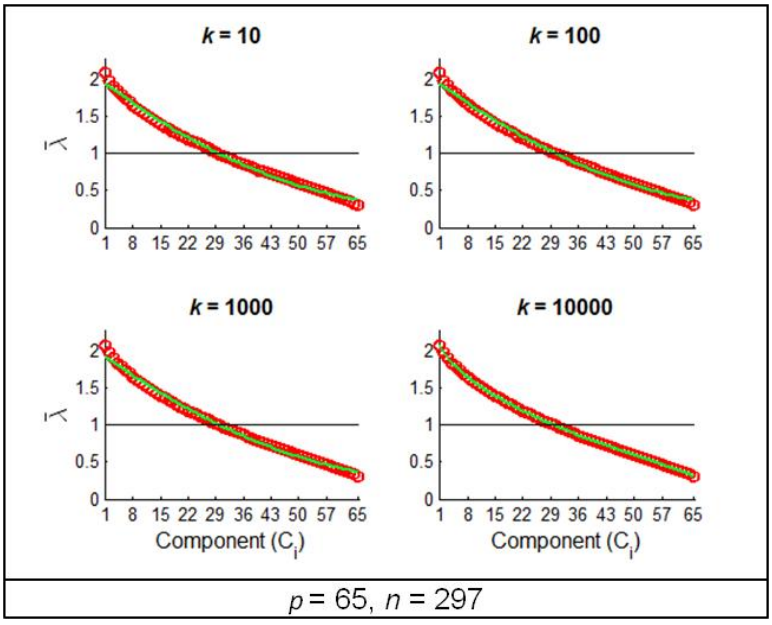


Figure 3.23. Subplots of Horn's curves produced from various k iterations of Monte Carlo simulations. Lines of red circles are MEVs, green lines are the 2OM fitted curves.

parameter *k* from which our existing data is constructed. In Figure 3.23 we see the results of the experiment. The overlapped green line on top of the red o's marks each eigenvalue along Horn's curve and compares the lines for any trends with respect to *k*. As viewed in the upper right subplot, $k = 100$ offers a satisfactory fit to the example dataset and while there are some small variations between the 2OM and MEV lines, there is no trend present that would be cause for alarm.

We also observe that $k = 10{,}000$ is the best fit in that the two lines follow the exact same path (this observation should be expected for such a high value of *k*.)

However, the difference between the two solutions is slight and does not merit a hundredfold increase in computation time. Based on these exploratory runs, $k = 100$ remains a suitable selection for purposes of linear regression least-squares second-order model fitting.

3.12.4. Model Adequacy

A standard and necessary procedure for linear regression model fitting is checking for basic assumptions (Montgomery, Peck, & Vining, 2006:122). They include:

- A linear relationship exists between the response and predictor;
- The error term ε has zero mean;
- The error term ε has constant variance σ^2;
- The errors are uncorrelated; and
- The errors are normally distributed.

In the case of the 2OM, the response $\hat{\lambda}$ is a linearly independent (orthogonal) product of random variables sampled from the known $\sim NID(0, \mathbf{I}_p)$ population distribution defined

for the MCS. At no point is noisy, real-world data introduced into the MEV stream; therefore, the model adequacy assumptions are satisfied.

3.12.5. Nearest Neighbor Interpolation for the 2OM

The lookup table has been reduced to five columns but still needs to be searched during a (p', n') query for `nnlp`, `nnup`, `nnln`, and `nnun`. The methodology discussed earlier for searching and sorting NNs (Section 3.10) and interpolating (Section 3.11) has not changed; the only difference is fewer columns of data need to be organized (each row in the coefficients lookup table is five columns wide). New methodology employed.

3.12.6. Random Data Graphs Comparisons

There is motivation to compare visually the 2OM graphs to those produced by the original lookup table. Of concern are "Was any accuracy lost for the reduction of lookup table size?" and "Does the curve fitting and least-squares introduce variation to the method?" The parsimony in the 2OM is not worth risking the accuracy already available to us in the MEV. Fortunately, the answer to both questions is "No."

Visual analysis of side-by-side comparisons of the graphics indicates performance is similar for each strategy. In Figures 3.24 - 3.26, the 2OM figures are in Panel A and the MEV ones are in Panel B. The largest difference appears along the horizontal axis. Because the 2OM can evaluate each curve at a number of points equal to p', the horizontal axis in the 2OM figures extend to p' and is not truncated at $C_{p^{(-)}}$ (as it is for MEVs during the NN search). Additionally, the 2OM curves appear to be smoother. In comparison, the MEVs are plotted in a ‘connect-the-dots’ fashion with no algebraic computation of intermediate values.

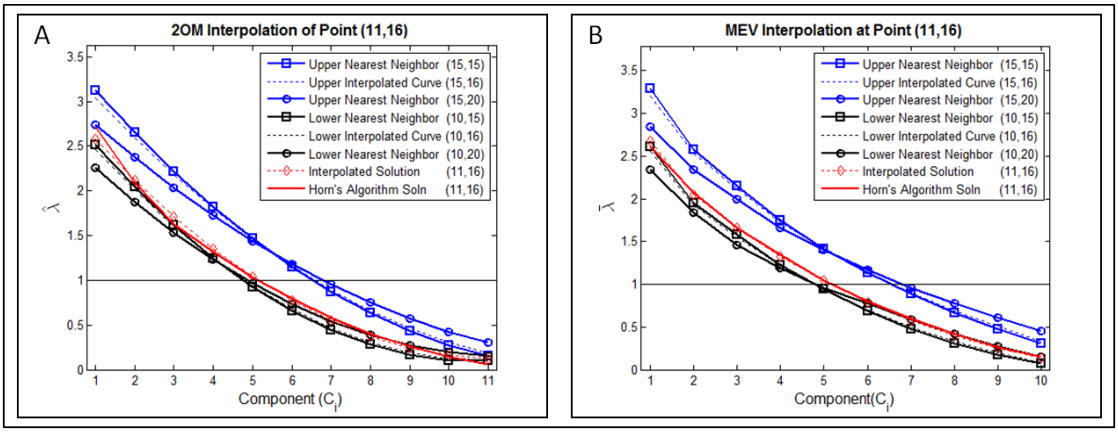


Figure 3.24. Visual comparison of results for a very small dataset (11,16).

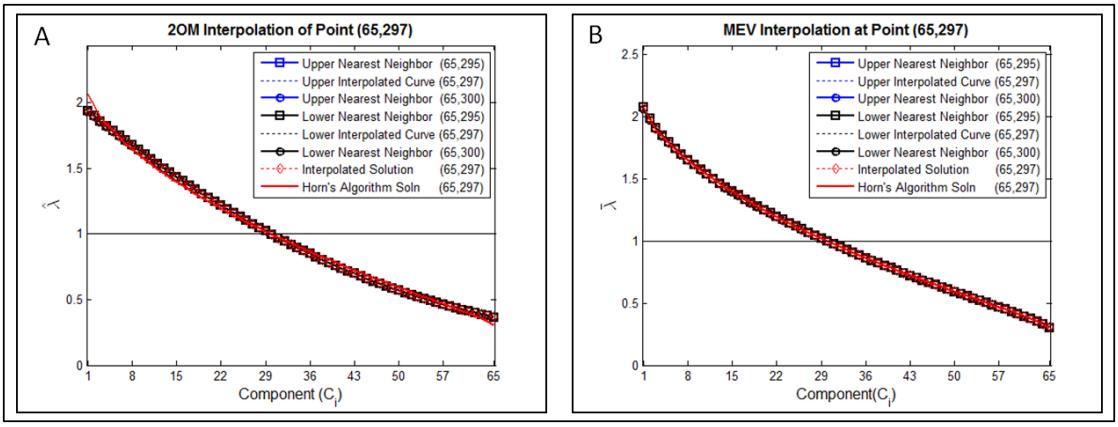


Figure 3.25. Visual comparison of results for moderate data size (65, 297).

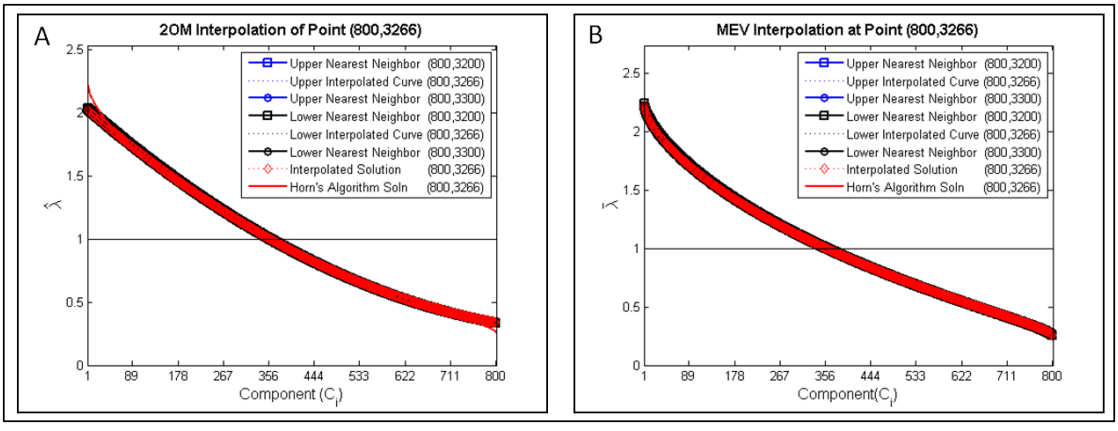


Figure 3.26. Visual comparison of results for larger data size (800, 3266).

In each strategy the known solution ('the truth curve'; an application of Horn's algorithm `EigenMean` from Figure 3.3 for direct computation of the MEVs describing the coordinate pair) is calculated and overlaid into each figure as a heavy, solid red line. In nearly every instance, the interpolated (p', n') Horn's curve is indistinguishable.

The visual agreement between solution strategies and within the interpolation routine indicates valid results are being produced. Therefore, the methodology developed thus far is capable of producing final results and we are ready to see how the final solution for estimation Horn's curve comes together in Chapter IV.

3.13. Methodology Summary

In this chapter, a technical examination of Horn's paper was conducted first to understand the method and then to build an algorithm capable of displaying Horn's curve. The individual elements of the algorithm are:

- Monte Carlo simulation generation of random data;
- Creating correlation matrix of the random data;
- Eigendecomposition of the correlation matrix;
- Indexing, averaging, and storing the MEVs in the lookup table;
- Searching the lookup table for nearest neighbors $p^{(+)}$, $p^{(-)}$, $n^{(+)}$, $n^{(-)}$ points according to one of four cases, depending upon the location of (p', n') in the lookup table;
- Interpolation of nearest neighbors to produce surrogate data before in-turn interpolating the surrogate data for estimated MEVs; and
- Plotting of the estimated MEVs to create Horn's curve for (p', n').

Once the MEV-based algorithm was complete, various sized (p', n') pairs were selected and used to query the lookup table. The interpolated curves were overlaid with a direct computation of Horn's curve of (p', n').

After MEV algorithm functional integrity was verified, the next step consisted of refining the lookup table data into a table of linear regression second-order coefficients. The second-order model algorithm permitted an 80-fold reduction in lookup table size with no loss of graphical accuracy in the completed Horn's curves. Visual analysis verified both algorithms perform as intended and have comparable results to each other. Our original goal is to complete Horn's test for an estimate of dimensionality for an n x p sampled dataset within range of the lookup table. In Chapter IV we will test the algorithms using sampled data from real-world experiments and produce the research objective of this thesis: An accurate stopping rule to produce a determination of multivariate data dimensionality using an estimate of Horn's curve.

IV. Results and Analysis

4.1. Chapter Overview

This chapter extends the exploratory analysis work done earlier to solving real-world problems. The objective is practical application; to make a contribution to practitioners wishing to solve principal components-type problems. To summarize our problem statement: Integral to successful PCA is determining when to stop extracting components – the matter is not a trivial one. Our solution – the goal – is to make Horn's test easier to use, meaning "with minimal time and effort." The large amount of random data needed has been preprocessed into manageable, nearly instantaneously available form, and algorithms have been written to produce an answer. The final link to a useful solution is bringing the *theory* to the *application* and *synthesizing them*.

4.2. Sampled Data Source

Thus far we have experimented with random data of known size. We surveyed the literature review to see *where* and *how large* typical studies might be but the actual *what* from a published database has not been used until now. Revisiting the UCI website, eleven datasets were selected as 'test subjects.' The reasons for selecting these particular ones are many: The type of data they contain (regression or classification), how much conditioning of the data was needed (non-numeric characters, missing values, NaN, Inf, non-invertible are all no-gos), a representative sample for the lookup table (right-sized n x p). As much as possible of the original data was kept; editing and conditioning was kept out to a minimum. Even so, the findings and evaluations given here may differ from other studies accomplished with the same data. Table 5.1 lists the datasets.

4.3. Putting It All Together

We have sufficient and necessary components to marry the theory with application. In Chapter III two flowcharts were used to describe how the two solution strategies function individually – Figure 3.2 for random data and Figure 3.10 for sampled data. Accompanying MATLAB scripts, one per each flowchart, details a body of functional code (Figure 3.3 and Figure 3.11, respectively). However, the two parts are not much use individually; the conjunction is required to produce *the* solution for an accurate estimation of how many components to extract for PCA.

4.4. Running of the Main MATLAB Script for The Mean Eigenvalues Approach

Before proceeding to the visual results of Horn's curve, we first complete a progress check to verify functionality among the search, interpolation, and curve producing subroutines. The main program script is called `HornsCurveSampled.m`. and there are variants for each the MEV and 2OM. The only difference is how the different dimensions of the two lookup tables are handled.

When this program is run, we are given a choice of multivariate studies in the directory and also presented an option to load one under another name. In our example case, we choose option '`1`' for the `Forest Fires` dataset. The main program loads the lookup table, determines the size of the lookup table, retrieves the user-requested file, loads the data matrix, and then sweeps the data for size requirements (has to be within range of **T**) and ensures it is not underdetermined ($p > n$). If a problem is found, the user is notified what the problem is and given a chance to either reload another file or quit. See Figure 4.1 for an image of the user input screen/menu.

```
This script will estimate Horn's Curve to aid in making a
Principal Components Analysis (PCA) dimensionality deter-
mination for an actual--sampled--data set.  Horn's Curve
is found by interpolating known, "ideal" data of size
equivalent to the actual sample size.  Constraints regarding
input and what the script can do are listed below.
The input values must be within these ranges:

  # of variables (p)    --> {5,1000}
  # of observations (n) --> {5,7000}

A crucial condition to consider is underdetermined data; that is,
data having fewer observations n than features p.  PCA of
underdetermined data is possible; however, this script does not
accept such datasets.

Please choose a dataset to load.  Type the number and press
'Enter.'  If the dataset is not listed, choose '0' (zero) and type
in the filename.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
(1)  Forest Fires
(2)  Glass
(3)  Parkinsons
(4)  SECOM
(5)  Seeds
(6)  Semeion
(7)  Steel Plates
(8)  Wisconsin Breast Cancer Study
(9)  Wines (Set 1)
(10) Wines (Set 2)
.....................................................
(0)  Manually enter a filename
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-> Please make a selection (1-10) or (0):
```

Figure 4.1. Main program user interface.

Once the data preliminaries check out, the algorithm assigns the number of rows of the matrix to n' and the number of columns to p'. Control is then passed to the function `findcurves.m` which is the search algorithm used to find the nearest neighbors pairs in the lookup table, interpolate the surrogate curves, and calculate the estimated Horn's curve. The interpolated curve is passed back and `HornsCurveSampled.m` again has control over program flow. Next, the dataset

undergoes correlation, eigendecomposition, and sorting of the eigenvalues (Figures 3.10 and 3.11; recall that there is no averaging of sampled data eigenvalues).

At this point, the merging of the real-world data (the user specified file) and idealized data (the interpolated solution) occurs and Horn's test takes place. Points along the sampled curve and the estimate of Horn's curve are checked component by component. These are the cases that may be encountered and the outcomes:

- If the sampled data (plotted in the scree line) is larger than Horn's curve at a component, then that component is considered significant and it should be extracted for analysis.
- If the scree line falls below Horn's curve, then those components are considered insignificant and may be discarded.
- If a component is below Horn's curve but above Kaiser's criterion at $\lambda = 1.0$, then it is considered *contested*. Contested points should be further evaluated by the analyst for significance to the study at hand.

At this time we can get information about how much variance the components in each of these cases is representing. This is vital for PCA since the summarization of variance per component and the cumulative amount of variance the dimensionality estimate retains is of value to the practitioner. The variance information will not be shown on the graph because it is too unwieldy; rather, it will be displayed in the MATLAB Command Window and stored in vector format (located in the variable Workspace) should the analyst want it.

4.4.1. Figure Output and Visual Analysis

The two components are stitched together to produce one figure displaying the estimate of Horn's curve and the results of Horn's test. A quick visual shows what the individual pieces looked like and how they come together in the solution.

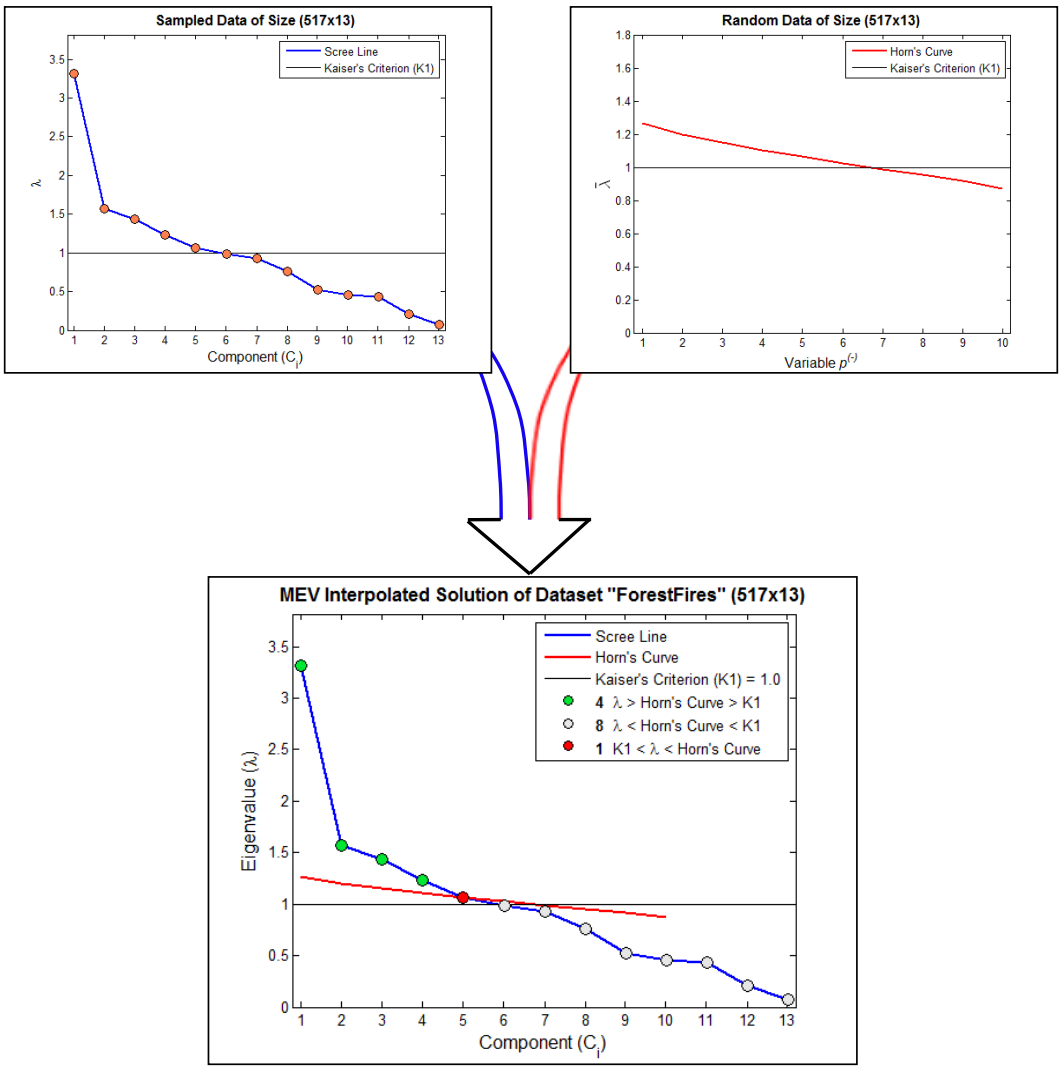


Figure 4.2. Conjunction of sampled and random data components in the finished product using the interpolated mean eigenvalue (MEV) solution of Horn's test.

In Figure 4.2, we see the solution (large center graphic) is a composite of the sampled data (upper left) and random data (upper right) halves. The solution includes not only an estimation of Horn's curve, it also features two common visual elements of PCA – Kaiser's criterion and the scree line.

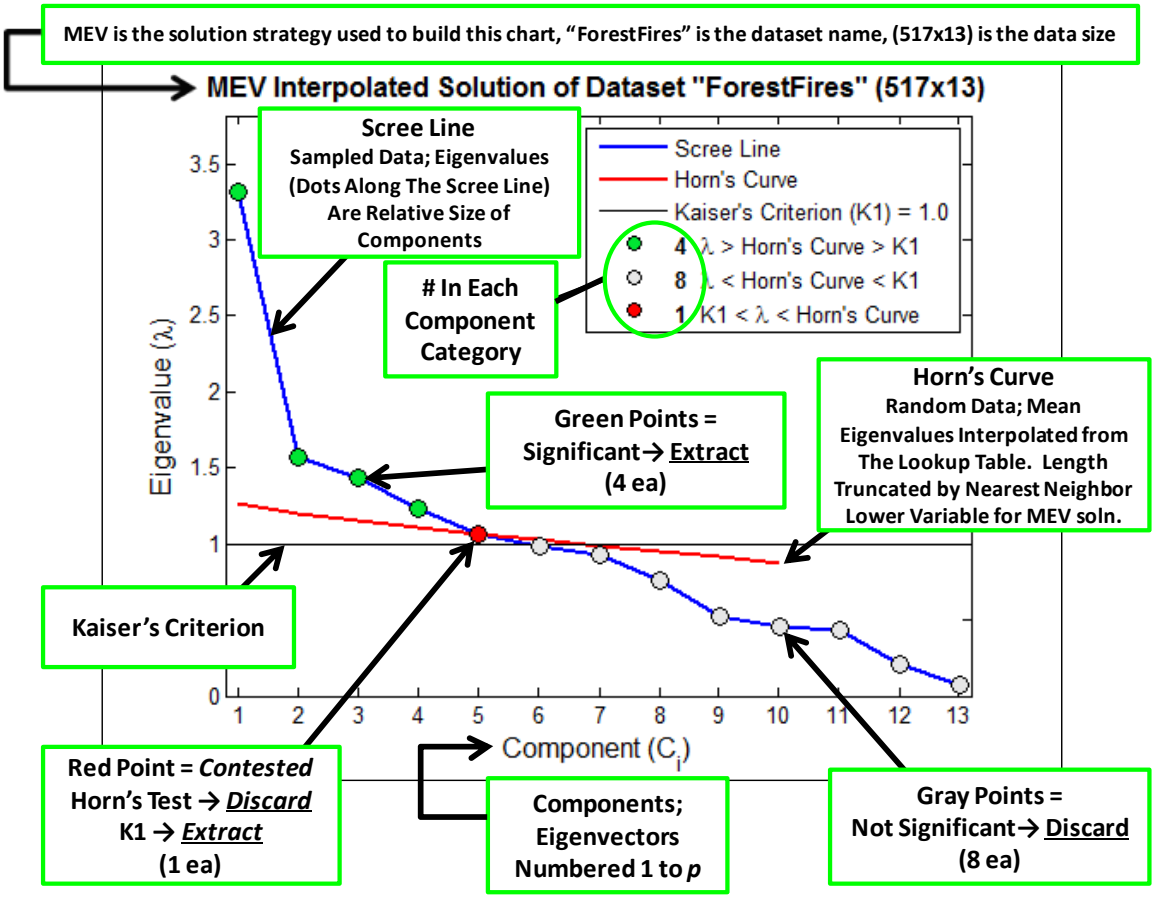


Figure 4.3. Detailed description of the interpolated solution of Horn's test.

Figure 4.3 provides a detailed explanation of each feature in the graphic and how to interpret them. The legend box contains a count of the components in each category. In this example, there are four components that are likely candidates for extraction (green points), eight that could be discarded due to low explanation of total variance (grey points), and one component that is contested (red point).

The figure is designed to be a stand-alone analysis product; that is, it can be shared electronically using any common picture file formats (.JPG, .PNG, .BMP, etc.) For this reason, the algorithm used to produce the result, the dataset name, and the data size are all included in the figure title. Referring again to Figure 4.3, the example shown is "MEV Interpolated", "ForestFires", and "(517x13)", respectively.

4.4.2. Components Dimensionality and Variation Summary Output

Horn's test is a visual analysis tool to aid in determining dimensionality but it is only one tool. Because PCA is a variance-oriented technique, understanding how the variance is distributed among the components provides the analyst with more information, often leading to better solution options for decision makers. The final step in completing Horn's test is to read back the variance dispersion information captured during evaluation of each component in relation to Horn's curve. This is done in the MATLAB Command Window and appears below the main program user interface (shown previously in Figure 4.1).

Figure 4.4 is a summary of the dimensionality assessment. The dimensionality estimate is equal to the number of components that meet Horn's test criteria for extraction. This is the actionable part of the analysis and is the solution to the problem statement. Following the dimensionality estimation, we can determine how the variance is spread among the components. In the `Forest Fires` data, one component (C_5) is contested and its eigenvalue is 1.0637.

```
-> Please make a selection (1-10) or (0): 1
Getting eigenvalues of (517x13) sampled data...Done!
Plotting all curves...Done!
************************* Summary ****************************
Dimensionality is estimated at 4 principal components by
Horn's test.  There are a total of 1 eigenvalues below Horn's
curve and greater than K1.  These eigenvalues should be further
evaluated against additional criteria for usefulness.
(Additional criteria == qualitative and quantitative
aspects of the study that are particular to a dataset,
purpose of the study, and analyst selection.)

-->Component #: 5  Eigenvalue: 1.0637 <--

Proportion of total variance explained by Horn's = 58.11%.
Additional proportion of total variance explained by the
contested "between the curves" components: 8.18%.
If 1 contested components are included, proportion = 66.29%.
**************************************************************
THE FILENAME USED IN THIS ANALYSIS IS ForestFires

End of processing.
```

Figure 4.4. Components dimensionality and variance summary output.

By applying Equation (3.8) (we do not specify a target variance T in this case) to the $j = 4$ principal components in the dimensionality assessment, we get a variance proportion of

$$\frac{1}{p}\sum_{i=1}^{j}\lambda_i \cong T,\ j \le p : i, j, p \in \mathbb{N}\ (0 \notin \mathbb{N})$$

$$\frac{1}{p}\sum_{i=1}^{4}\lambda_i = \frac{1}{p}\left(\lambda_1+\lambda_2+\lambda_3+\lambda_4\right) = \frac{1}{13}\left(3.3153+1.5693+1.4369+1.2323\right) = \frac{7.5538}{13}$$
$$= 0.58106$$

which is the value given for "Proportion of total variance explained by Horn's" in Figure 4.4.

Below that, in the next two lines, is the result of applying Equation (3.7).

$$\frac{\lambda_i}{p},\ i \le p : i, p \in \mathbb{N}\ (0 \notin \mathbb{N})$$

$$\frac{\lambda_5}{p} = \frac{1.0637}{13} = 0.08182$$

If the two findings (Horn's test and any contested components) are combined, Equation (3.8) is reevaluated for significant and contest components and the proportion of variance explained by the five components is 0.6629, or 66.29%. A gain of 8.18% variance might be significant to the analyst, and if so, then the number of principal components equals five. The feature of tabulated variance permits some flexibility in the analyst's assessment.

Included as information to the user are the top two lines (above the `*Summary*` block); they occur during script execution just to show the program has not stalled in a routine. For reference, the user can see what size the data is without having the figure (such as Figure 4.3) visible. The last couple lines of the screen output is a read back of the filename used for the analysis and notification that the script has completed execution and has stopped processing.

4.5. Running of The Second-Order Model Script

We have just seen how to select a dataset and interpret the visual and summary results for the mean eigenvalue solution. What about the linear regression second-order model (2OM) and the table of coefficients?

The two approaches use an identical user interface for input and output. There are a few subtle differences within the code, mostly due to what lookup table and how the mean eigenvalues are determined; otherwise, each version shares the nearest neighbors

search, interpolation of surrogate curves, and data plotting routines. To distinguish the graphical results, the titles of every figure include what type of solution was used to produce it (MEV or 2OM).

We conclude this section with an example of output from the 2OM solution approach. It is shown here in Figure 4.5 using the `Forest Fires` data from Figure 4.5, this time without the bumper stickers.

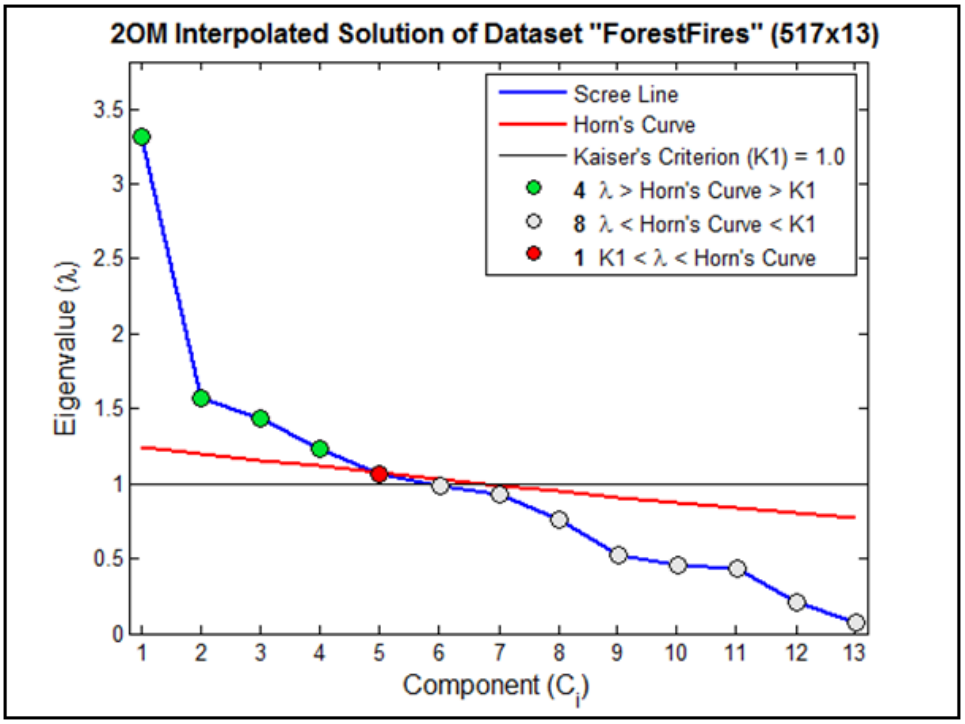


Figure 4.5. Interpolated second-order model (2OM) solution of Horn's test of the ForestFires dataset. Details are similar to those found in Figure 4.3.

4.6. Challenges

Constructing an algorithm that handles data of different sizes is not problematic; handling data within stated constraints (min/max bounds in the lookup table, no underdetermined sets in this case) requires more methodical planning and logic in the scripts but is still controllable. The greater unknown is the format of the data existing in

the real-world: Weak or strong correlation, multicollinearity, non-constant variance, different probability distributions are all valid concerns. A perfect solution does not exist; a calculated manner in which these issues are handled is a reasonable goal. An attempt was made to stress the algorithms presented herein using a representative sample of both size and variety of data. Certainly not every possible configuration of data was presented to the algorithm for testing. Therefore, the author anticipates that unexpected results in the future may be a possibility due to the behavior of the eigenvalues affected by characteristics of the sample data. Part of being an analyst is discovery through exploration; situations just described are opportunities to investigate underlying causes.

4.6.1. Lookup Table Size

Presently, the limiting factor in application is the size of the lookup table. The only way to increase its size is to run Monte Carlo simulation on unmapped (p, n) pairs in the study region. As the data size increases, particularly as the number of variables grows, it will take much longer to add each new row to the lookup table. Expanding the (p, n) region-of-interest is possible and is made more attractive given that the 2OM is comparable in performance to the MEV. It is straight-forward to process the least-squares fitted coefficients and augment the lookup table. The implication is the table of mean eigenvalues will not advance beyond what currently is in the application, but given the ancillary goal of creating as small a data footprint as possible, the omission of future entries to **T** is plausible.

4.6.2. Software Required

Not every computer runs MATLAB and not every practitioner is well-versed in MATLAB usage. As such, the target audience is presently only MATLAB users. Fortunately, MATLAB output is used easily in other applications:

- Graphs can be shared as metafiles or a number of picture formats and included as objects in MS Word® and MS PowerPoint®.
- All Command Window text can be copied and pasted as editable text and MATLAB variables can be copied from the Workspace and moved to MS Excel® for editing in a spreadsheet (the opposite is also true).
- A low-tech but workable approach is to copy and paste the Horn's test data into an Excel spreadsheet and use the graphing capability of Excel to reproduce the figure in MS Office®.

A desirable solution is cross-product porting of the script to Java® or MS Office. The MathWorks produces a free compiler and packager called MCR (MATLAB Compiler Runtime). This author has not worked with MCR but it appears to offer an excellent way to share .m files among users who do not have access to MATLAB.

4.7. Chapter Summary

The results achieved quite satisfactory and meet the stated research objective: To develop an accurate tool for determining the number of components to retain. The additional objectives of automating the tool to remove unwarranted subjective evaluation of the results were also reached. Additionally, the non-primary objectives of incorporating common visual elements of PCA stopping rules (the scree line and Kaiser's

criterion) to assist the practitioner were also met.

Finally, the MATLAB user interface developed makes easy work of loading data and then provides summary results for the dimensionality estimate and any contested components (that is, Horn's test and Kaiser's criterion arrive at different conclusions). The user is provided information regarding the total amount of original variance explained by the dimensionality assessment and, if there are contested components, what additional variance the contested components represents.

Comparisons of both solution algorithms are thus far identical in both visual analysis and variance summary findings. The side-by-side results of each comparison are not included in this chapter; please see Appendix I: Results for Sampled Datasets. For a line-by-line list of all the computer code leading to a result described by this thesis, please see Appendix II: MATLAB Scripts.

V. Discussion

5.1. Relevance of the Current Investigation

Automation of Horn's method provides a powerful tool for PCA. During the literature review, several authors published findings in regard to the accuracy of various component extraction stopping rules: Horn's technique received remarks verifying a high level of component identification accuracy.

In contrast, the most widely-used stopping rule is Kaiser's K1 criterion: Retain components with an eigenvalue greater than or equal to one, discard those less than that. A simple analogy for this thinking (and there are certainly others) is one would not read a book and then write a lengthier summary than the book is long, so why keep a factor that has less summarizing power than the variable it is meant to transform? This is where science bows to art; the analyst is responsible to his practice to make an informed decision about the purpose of the analysis. There *are* qualitative aspects that have to augment all of these stopping rules.

5.2. Conclusions of Research

The author of this thesis is of the opinion why Horn's procedure is not used more often is it requires more preparatory work by the analyst and, to this author's knowledge, popular statistics software packages do not offer direct computation of it. Lack of understanding regarding the black box nature of specialized commercial software leaves one at a disadvantage when unique challenges require unique solution strategies – if the only tool in the tool shed is a hammer, suddenly all the problems appear to have nails for solutions. It is likely not everyone has the time, skill, or impetus to pursue application of

Horn's test for an individual problem. It is hoped that the work shared in this thesis permits others to gain insight into multivariate analysis solution techniques they might not previously been inclined to explore.

5.3. Limitations

Only eleven datasets available in the public domain were tested and it is possible that data of unknown configurations could present pathological scree lines. One such case (and there are likely others) is a scree line that resembles a sideways view of a set of sloping steps that may hop back and forth across Horn's curve. Such a dataset probably exists – it meets the stated assumptions for *k*, is not underdetermined, is monotonically decreasing – and will present dimensionality results that have not yet been considered.

It is also anticipated a case exists in which the MEV and 2OM solutions may disagree in their conclusions; that is, each algorithm presents a different estimate of dimensionality. For instance, datasets featuring shallow intersection angles between Horn's curve and the scree line (i.e., almost parallel along some interval of components) will likely to lead to under extraction of components by the 2OM. Since under extraction discards information, it is this author's recommendation that, should this situation be encountered, the MEV strategy be used to verify the 2OM dimensionality assessment.

5.4. Future Work/Further Research

1) Confirmatory analysis of the accuracy of the Horn's test algorithm should be an immediate next step. This can be accomplished by structuring of random data with known dimensionality and then presenting it to the algorithm as a sampled dataset.

2) Creation of a routine to handle limited problems beyond what the lookup table can immediately reference. A dry run on a real-world dataset of 112,000+ observations on 121 variables was successfully completed in under five minutes of processing time. There is certainly room to expand in solving small *p*, large *n* problems without adverse expenditure of computer resources while analyses await.

3) Combining the MATLAB script with a graphical user interface capable of giving easier access to the results is desirable.

4) Artificial neural networks present possibilities to learn the region-of-interest. If so, not only can estimates of dimensionality be determined for (p', n') but the need to have ready the (p, n) lookup table is eliminated.

5) Principal components analysis is a gateway to other multivariate analysis techniques. Expanding the code, or modularizing it, so that other methods (specifically factor analysis) can access the dimensionality estimation extends application.

Appendix I: Results for Sampled Datasets

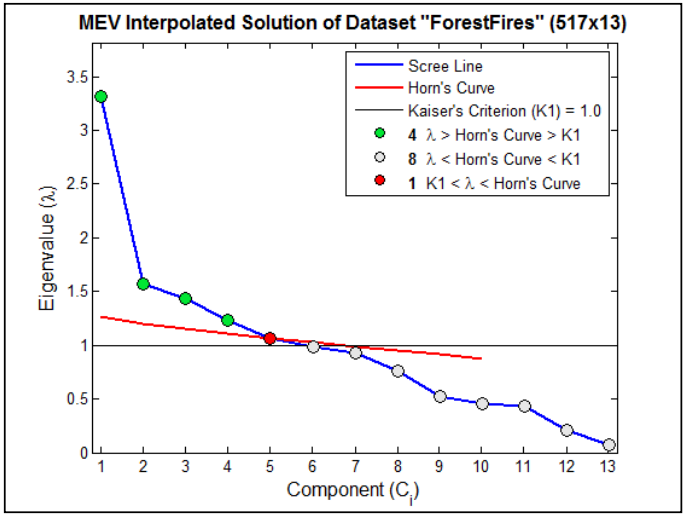
The University of California-Irvine Center for Machine Learning and Intelligent Systems data repository (http://archive.ics.uci.edu/ml/) was the source used for real-world data. Without the resources of the UCI website, this thesis would have been limited in scope.

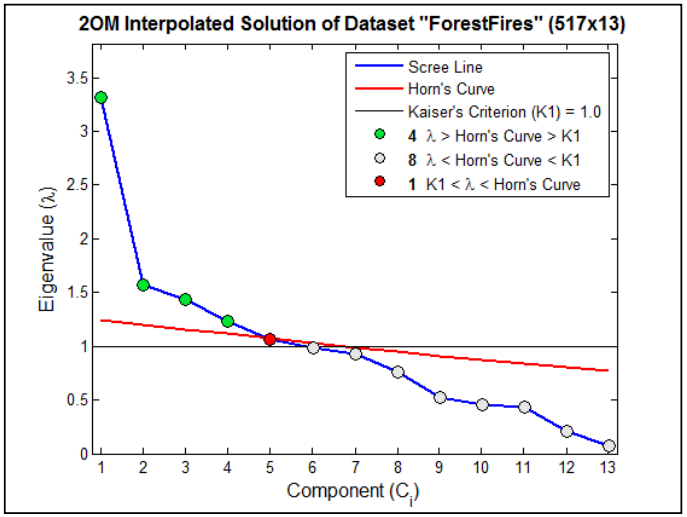
During the course of experimentation, some data would not complete eigendecomposition due to NaN, Inf, or non-numerical data types (missing entries or string characters). Trimming of data occurred at the *minimum level* necessary to achieve functionality. Wherever possible, rows (observations) were deleted in lieu of columns (variables). As such, these truncations may result in smaller sizes of the named datasets than from what is found elsewhere or used by researchers for other analyses.

The bibliography lists contributing donors or the stewards of such data; however, in an effort to invite additional exploration, the URLs of each dataset is given here:

Table 5.1. Web addresses of each dataset used to test the algorithms.

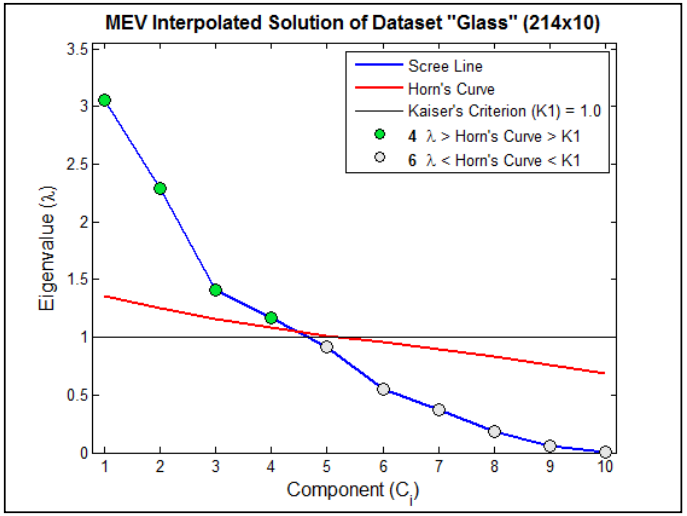
Dataset	*Web Address (URL)*
Forest Fires	http://archive.ics.uci.edu/ml/datasets/Forest+Fires
Glass	http://archive.ics.uci.edu/ml/datasets/Glass+Identification
Parkinsons	http://archive.ics.uci.edu/ml/datasets/Parkinsons
SECOM	http://archive.ics.uci.edu/ml/datasets/SECOM
Seeds	http://archive.ics.uci.edu/ml/datasets/seeds
Semeion	http://archive.ics.uci.edu/ml/datasets/Semeion+Handwritten+Digit
Steel Plates	http://archive.ics.uci.edu/ml/datasets/Steel+Plates+Faults
WI Breast Cancer (Original)	http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29
Wines 1	http://archive.ics.uci.edu/ml/datasets/Wine
Wines 2	http://archive.ics.uci.edu/ml/datasets/Wine+Quality
Iris	http://archive.ics.uci.edu/ml/datasets/Iris

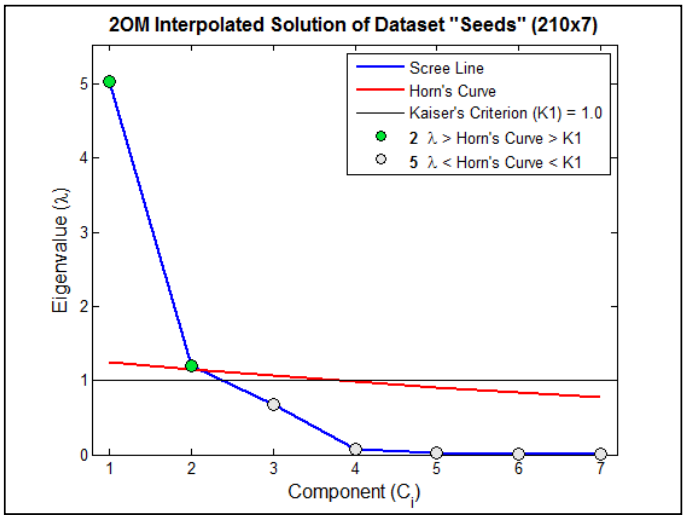





```
-> Please make a selection (1-10) or (0): 1
Getting eigenvalues of (517x13) sampled data...Done!
Plotting all curves...Done!
************************** Summary ******************************
Dimensionality is estimated at 4 principal components by
Horn's test.  There are a total of 1 eigenvalues below Horn's
curve and greater than K1.  These eigenvalues should be further
evaluated against additional criteria for usefulness.
(Additional criteria == qualitative and quantitative
aspects of the study that are particular to a dataset,
purpose of the study, and analyst selection.)

-->Component #: 5  Eigenvalue: 1.0637 <--

Proportion of total variance explained by Horn's = 58.11%.
Additional proportion of total variance explained by the
contested "between the curves" components: 8.18%.
If 1 contested components are included, proportion = 66.29%.
*****************************************************************
THE FILENAME USED IN THIS ANALYSIS IS ForestFires

End of processing.
```

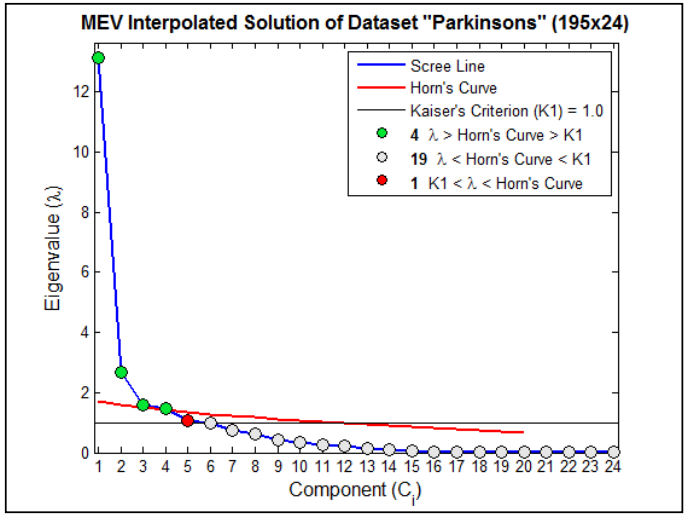
Figure AI01. Dataset Forest Fires (Cortez & Morais, 2007)

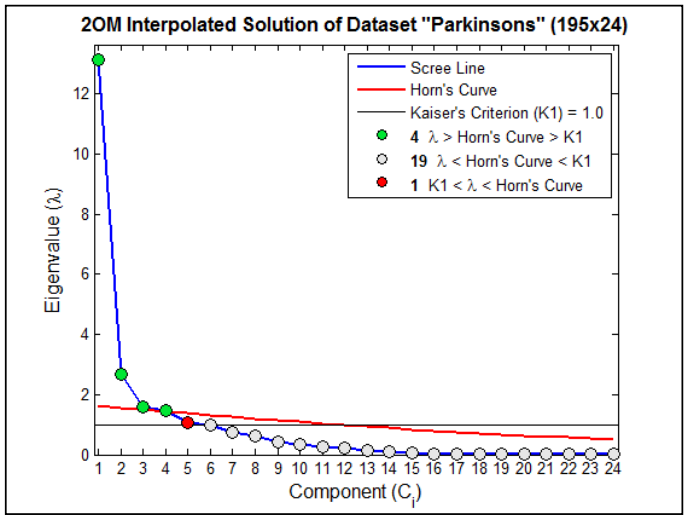





```
-> Please make a selection (1-10) or (0): 5
Getting eigenvalues of (210x7) sampled data...Done!
Plotting all curves...Done!
************************** Summary ******************************
Dimensionality is estimated at 2 principal components.
Proportion of total variance explained = 88.98%.
There are no contested components.
*****************************************************************
THE FILENAME USED IN THIS ANALYSIS IS Seeds

End of processing.
```

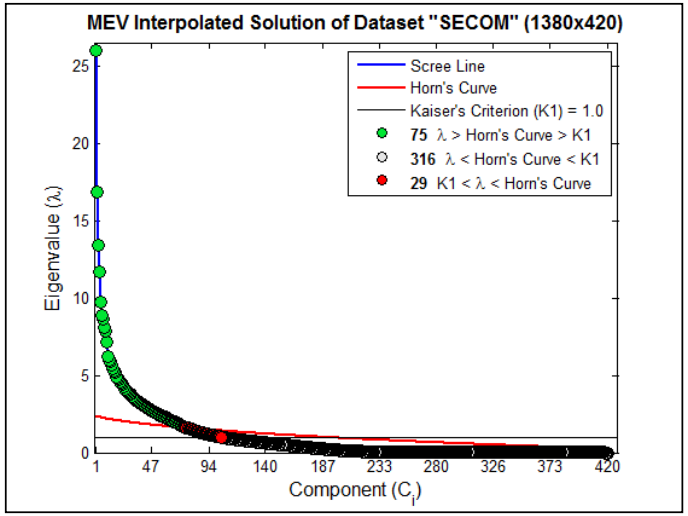
Figure AI02. Dataset Glass (Frank & Asuncion, 2010)

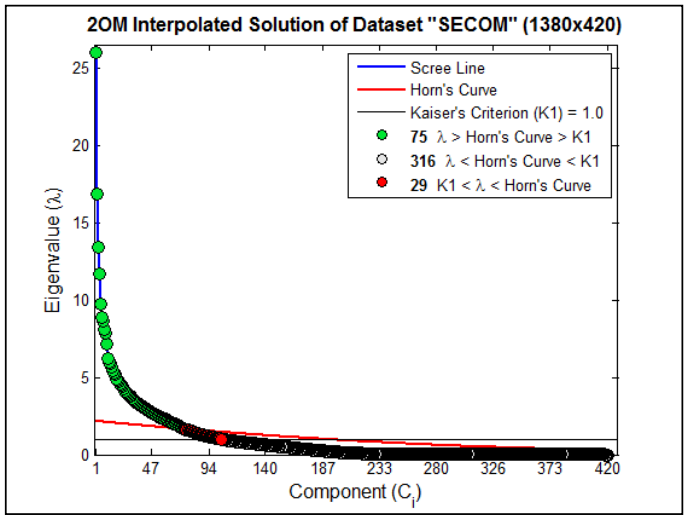





```
-> Please make a selection (1-10) or (0): 3
Getting eigenvalues of (195x24) sampled data...Done!
Plotting all curves...Done!
************************** Summary ******************************
Dimensionality is estimated at 4 principal components by
Horn's test.  There are a total of 1 eigenvalues below Horn's
curve and greater than K1.  These eigenvalues should be further
evaluated against additional criteria for usefulness.
(Additional criteria == qualitative and quantitative
aspects of the study that are particular to a dataset,
purpose of the study, and analyst selection.)

-->Component #: 5  Eigenvalue: 1.0657 <--

Proportion of total variance explained by Horn's = 78.73%.
Additional proportion of total variance explained by the
contested "between the curves" components: 4.44%.
If 1 contested components are included, proportion = 83.17%.
*****************************************************************
THE FILENAME USED IN THIS ANALYSIS IS Parkinsons

End of processing.
```

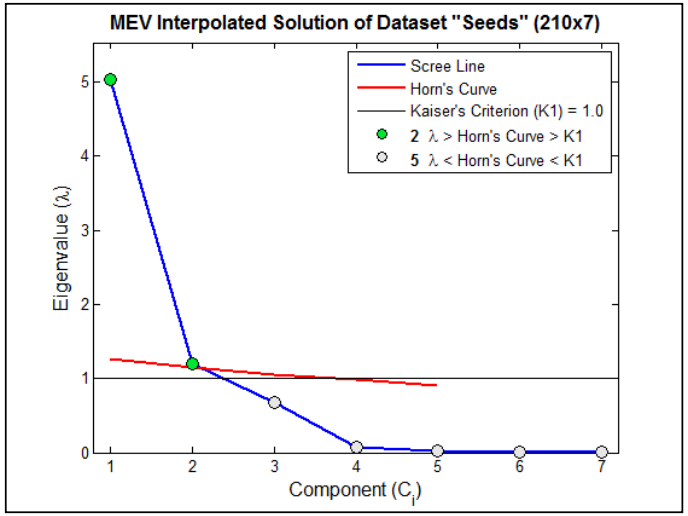
Figure AI03. Dataset Parkinsons (Little, McSharry, Roberts, Costello, & Moroz, 2007)

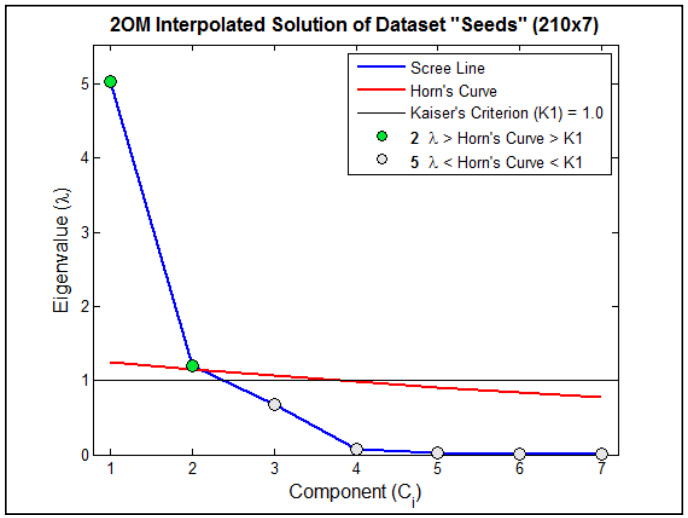





```
-> Please make a selection (1-10) or (0): 3
Getting eigenvalues of (195x24) sampled data...Done!
Plotting all curves...Done!
************************** Summary ******************************
Dimensionality is estimated at 4 principal components by
Horn's test.  There are a total of 1 eigenvalues below Horn's
curve and greater than K1.  These eigenvalues should be further
evaluated against additional criteria for usefulness.
(Additional criteria == qualitative and quantitative
aspects of the study that are particular to a dataset,
purpose of the study, and analyst selection.)

-->Component #: 5  Eigenvalue: 1.0657 <--

Proportion of total variance explained by Horn's = 78.73%.
Additional proportion of total variance explained by the
contested "between the curves" components: 4.44%.
If 1 contested components are included, proportion = 83.17%.
*****************************************************************
THE FILENAME USED IN THIS ANALYSIS IS Parkinsons

End of processing.
```

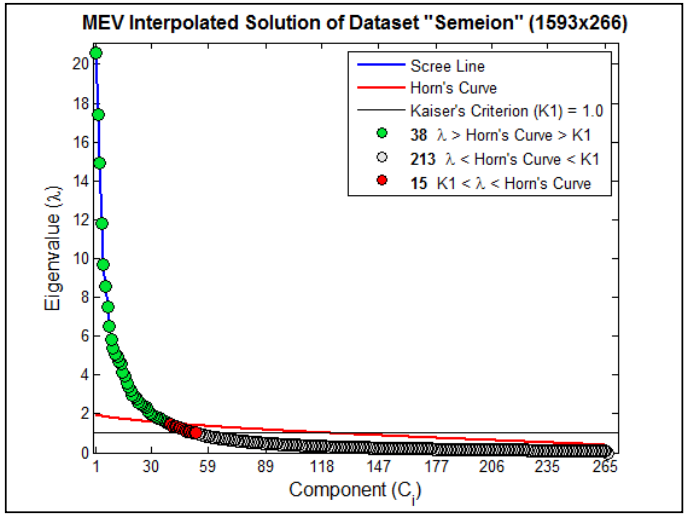
Figure AI04. Dataset SECOM (Frank & Asuncion, 2010)

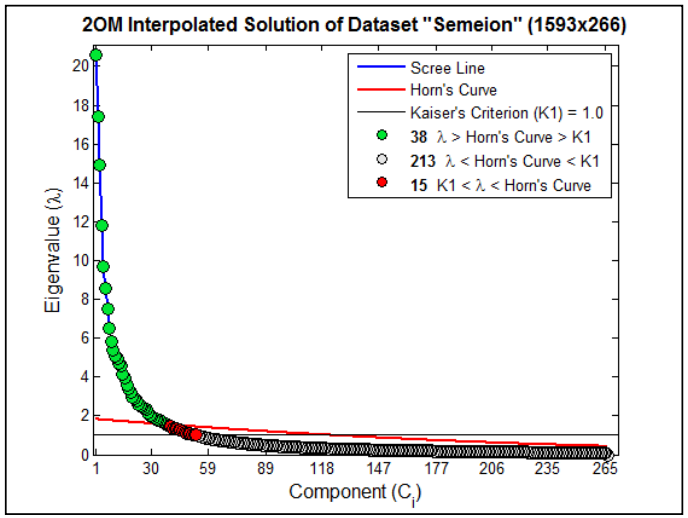





```
-> Please make a selection (1-10) or (0): 5
Getting eigenvalues of (210x7) sampled data...Done!
Plotting all curves...Done!
************************** Summary ******************************
Dimensionality is estimated at 2 principal components.
Proportion of total variance explained = 88.98%.
There are no contested components.
*****************************************************************
THE FILENAME USED IN THIS ANALYSIS IS Seeds

End of processing.
```

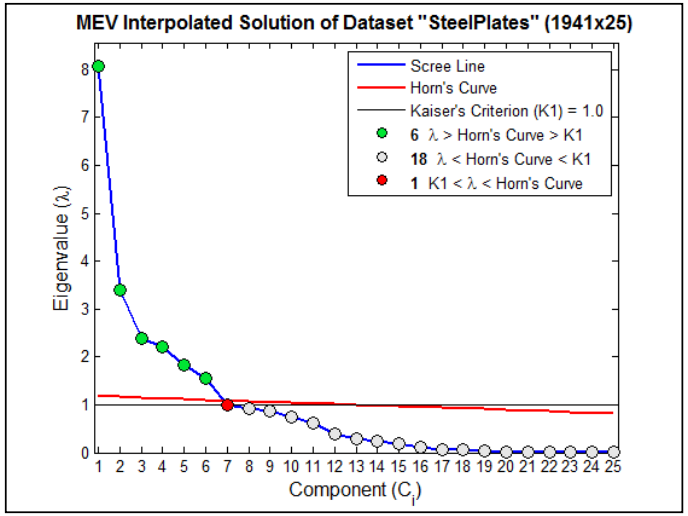
Figure AI05. Dataset Seeds (Kulczycki, Kowalski, Lukasik, & Zak, 2012) (Charytanowicz & Niewczas, 2012)

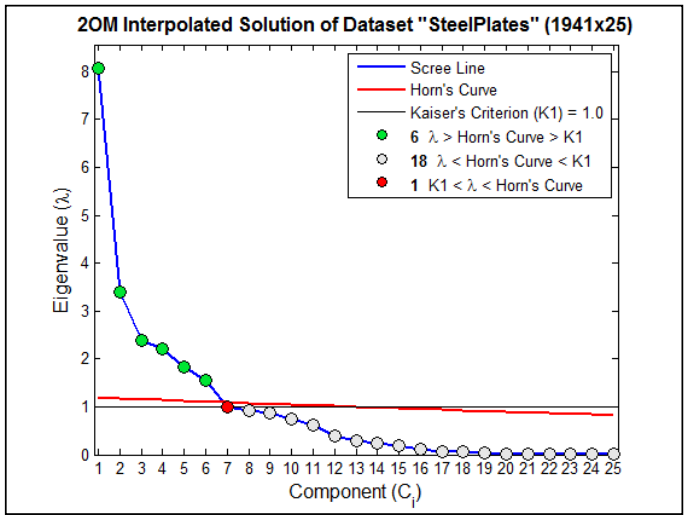





```
-> Please make a selection (1-10) or (0): 6
Getting eigenvalues of (1593x266) sampled data...Done!
Plotting all curves...Done!
*************************** Summary ******************************
Dimensionality is estimated at 38 principal components by
Horn's test.  There are a total of 15 eigenvalues below Horn's
curve and greater than K1.  These eigenvalues should be further
evaluated against additional criteria for usefulness.
(Additional criteria == qualitative and quantitative
aspects of the study that are particular to a dataset,
purpose of the study, and analyst selection.)

-->Component #: 39  Eigenvalue: 1.4770 <--
-->Component #: 40  Eigenvalue: 1.4516 <--
-->Component #: 41  Eigenvalue: 1.3701 <--
-->Component #: 42  Eigenvalue: 1.3398 <--
-->Component #: 43  Eigenvalue: 1.3192 <--
-->Component #: 44  Eigenvalue: 1.2618 <--
-->Component #: 45  Eigenvalue: 1.2463 <--
-->Component #: 46  Eigenvalue: 1.1977 <--
-->Component #: 47  Eigenvalue: 1.1289 <--
-->Component #: 48  Eigenvalue: 1.1103 <--
-->Component #: 49  Eigenvalue: 1.0926 <--
-->Component #: 50  Eigenvalue: 1.0761 <--
-->Component #: 51  Eigenvalue: 1.0497 <--
-->Component #: 52  Eigenvalue: 1.0218 <--
-->Component #: 53  Eigenvalue: 1.0073 <--

Proportion of total variance explained by Horn's = 70.17%.
Additional proportion of total variance explained by the
contested "between the curves" components: 6.82%.
If 15 contested components are included, proportion = 76.99%.
******************************************************************
THE FILENAME USED IN THIS ANALYSIS IS Semeion

End of processing.
```

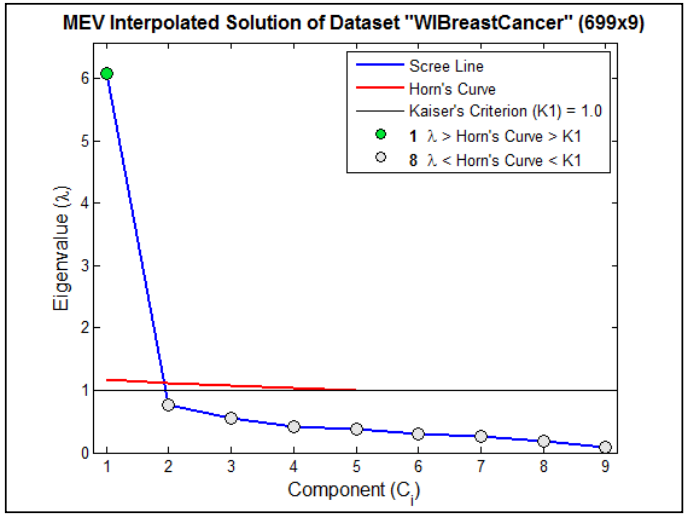
Figure AI06. Dataset Semeion Handwritten Digit (Semeion Research Center for the Science of Communication, 2008)






```
-> Please make a selection (1-10) or (0): 7
Getting eigenvalues of (1941x25) sampled data...Done!
Plotting all curves...Done!
************************** Summary ******************************
Dimensionality is estimated at 6 principal components by
Horn's test.  There are a total of 1 eigenvalues below Horn's
curve and greater than K1.  These eigenvalues should be further
evaluated against additional criteria for usefulness.
(Additional criteria == qualitative and quantitative
aspects of the study that are particular to a dataset,
purpose of the study, and analyst selection.)

-->Component #: 7  Eigenvalue: 1.0076 <--

Proportion of total variance explained by Horn's = 77.74%.
Additional proportion of total variance explained by the
contested "between the curves" components: 4.03%.
If 1 contested components are included, proportion = 81.77%.
*****************************************************************
THE FILENAME USED IN THIS ANALYSIS IS SteelPlates

End of processing.
```

Figure AI07. Dataset Steel Plates Faults (Semeion Research Center for the Science of Communication, 2010)




```
-> Please make a selection (1-10) or (0): 8
Getting eigenvalues of (699x9) sampled data...Done!
Plotting all curves...Done!
************************** Summary ******************************
Dimensionality is estimated at 1 principal components.
Proportion of total variance explained = 67.52%.
There are no contested components.
*****************************************************************
THE FILENAME USED IN THIS ANALYSIS IS WIBreastCancer

End of processing.
```

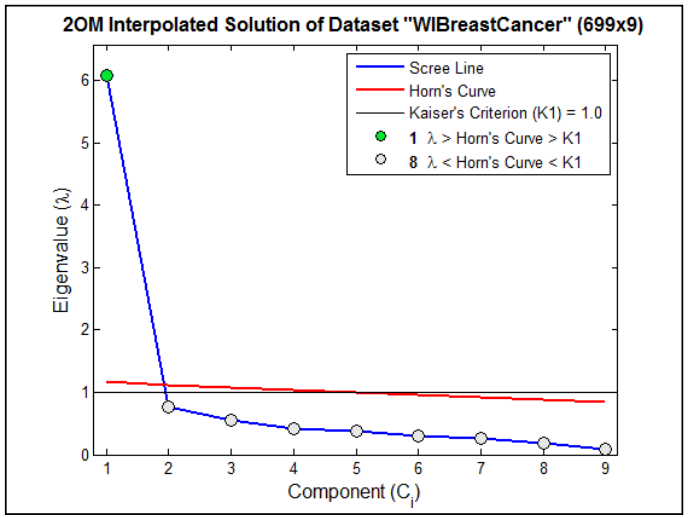
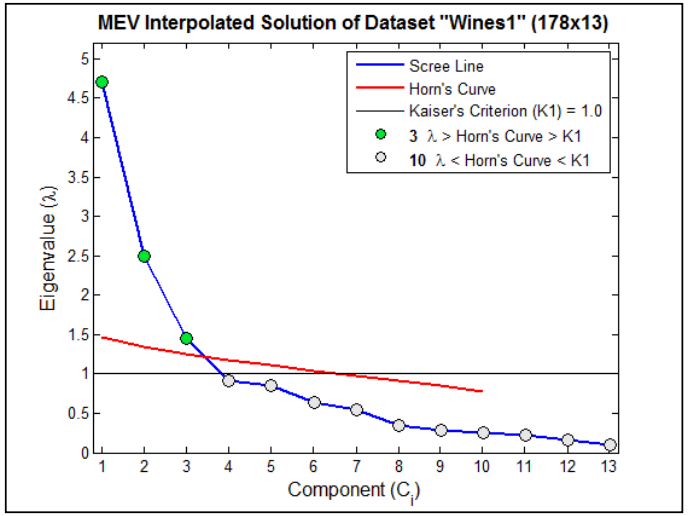


Figure AI08. Dataset Wisconsin Breast Cancer Study (Original) (Wolberg & Mangasarian, 1990) (Wolberg W. H., 1992)




```
-> Please make a selection (1-10) or (0): 10
Getting eigenvalues of (1599x11) sampled data...Done!
Plotting all curves...Done!
************************** Summary ******************************
Dimensionality is estimated at 4 principal components.
Proportion of total variance explained = 70.81%.
There are no contested components.
*****************************************************************
THE FILENAME USED IN THIS ANALYSIS IS Wines2

End of processing.
```

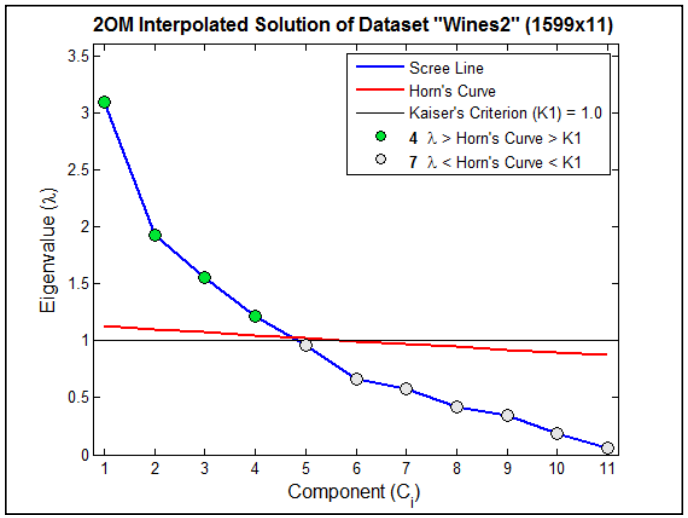
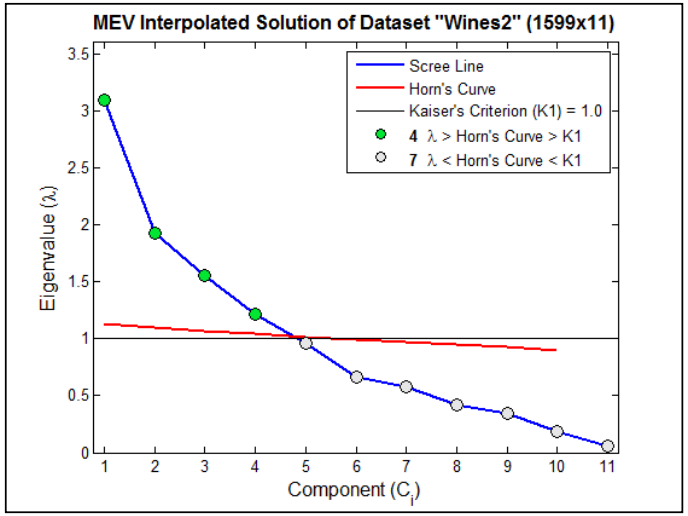
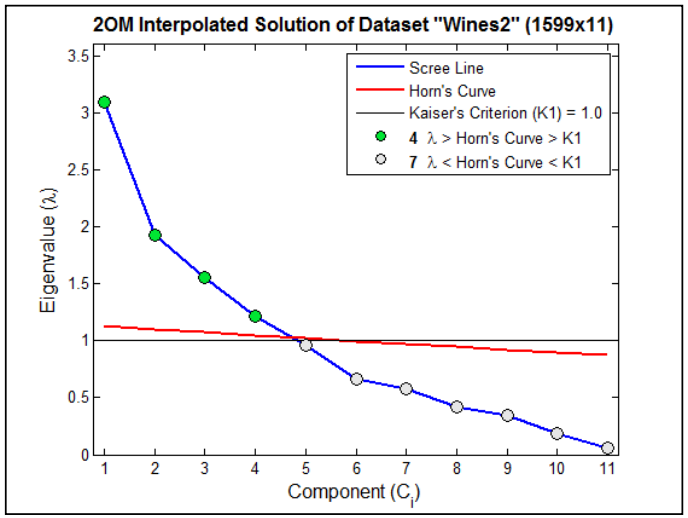


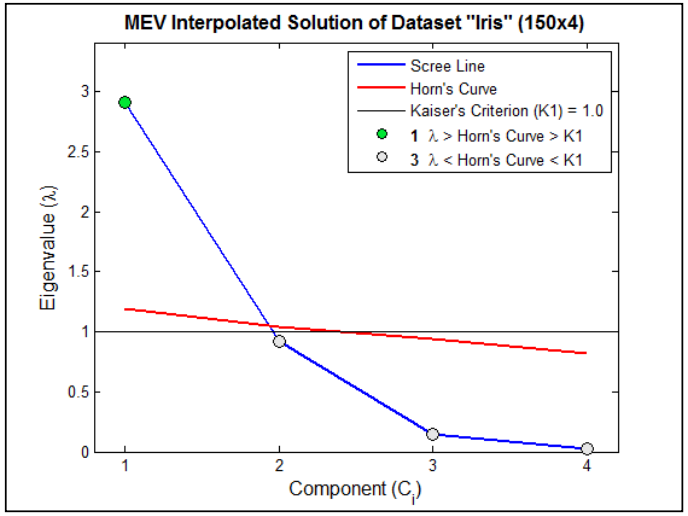
Figure AI09. Dataset Wines (Frank & Asuncion, 2010)






```
-> Please make a selection (1-10) or (0): 10
Getting eigenvalues of (1599x11) sampled data...Done!
Plotting all curves...Done!
************************** Summary ******************************
Dimensionality is estimated at 4 principal components.
Proportion of total variance explained = 70.81%.
There are no contested components.
*****************************************************************
THE FILENAME USED IN THIS ANALYSIS IS Wines2

End of processing.
```

Figure AI10. Dataset Wine Quality (Cortez, Cerdeira, Almeida, Matos, & Reis, 2009)




```
-> Please make a selection (1-10) or (0): 0
Please enter the filename (script assumes .mat)
--> Iris
The selected filename has data small enough for a direct
calculation of Horn's curve.

Getting eigenvalues of (150x4)...Done!
Plotting all curves...Done!
************************** Summary ******************************
Dimensionality is estimated at 1 principal components.
Proportion of total variance explained = 72.77%.
There are no contested components.
*****************************************************************
THE FILENAME USED IN THIS ANALYSIS IS Iris

End of processing.
```

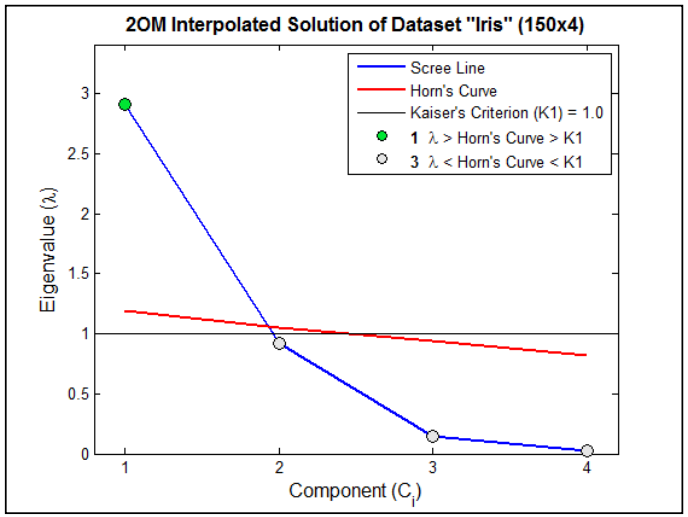


Figure AI11. Dataset Iris (Frank & Asuncion, 2010)

Appendix II: MATLAB Scripts

Main Script: HornsMethodRandomMEV.m

```
%Original code by Captain Andrew L. Bigley, USAF.  Written for partial
%fulfillment of a Master's of Science Degree in Operations Research, The
%Air Force Institute of Technology (AFIT), Wright-Patterson AFB, OH USA
%US Government disclaimer:
%The views expressed in this thesis are those of the author and do not
%reflect the official policy or position of the United States Air Force,
%Department of Defense, or the United States Government.
%This material is declared a work of the U.S. Government and is not subject
%to copyright protection in the United States.
%21 March 2013
%
%%Graph the eigenvalue curves for one coordinate pair.  This script
%references a table of pre-determined, sorted mean eigenvalues and
%then interpolates to trap the (p',n') pair in the information in the
%table.  Various configurations that may be presented by a user to the
%script are discussed below.
%
%initialize the workspace environment
close all; clear all; clc
%initialize global variables--they are in the datafile
global tablex ssizep ssizen;    %these variables are global in nature

load LookupTable.mat

maxp = max(tablex(:,1));    %the largest variable value in the data
minp = min(tablex(:,1));    %the smallest variable (a.o. 13 Jan minp=5)
                            %PCA on less than 5 variables?!?
maxn = max(tablex(:,2));    %largest # of observations in the mapped data
minn = min(tablex(:,2));    %smallest # of obs in the mapped data
cr = sprintf('\n');         %carriage return variable; use with 'disp'
%
%display opening message
disp(['This script will find Horn''s Curve as described by Monte  ' cr...
      'Carlo simulation generated and sorted mean eigenvalues.    ' cr...
      'This version evaluates random data of selected (p'',n'').  ' cr...
      'For use with actual, real world datasets, see script       ' cr...
      'HornsMethodSampled.']); cr;
disp(['The input values must be within these ranges:              ' cr...
      '                                                           ' ]);
fprintf('  # of variables (p)    --> {%d,%d} \n',minp,maxp)
fprintf('  # of observations (n) --> {%d,%d} \n',minn,maxn)
disp(['                                                           ' cr...
      'Also, make sure p is not less than n. ']); cr; cr;

%***check for input violations (out of range on p or n and n < p)***
%set datavalid flag to 'false'.  Assume the following:
paramvalid = false;     %valid input relation for p to n hasn't been rec'd
pvalid = false;         %valid variable input has not been entered
nvalid = false;         %valid observations input has not been entered
```

```
%
while paramvalid == false   %assume invalid parameters (relation of p,n)
    while pvalid == false   %assume invalid p value (relation to tablex)
        fprintf('Which variable (p) to graph?\n')
        getp = input('--> ');   %the variable of interest (pprime)
        if getp < minp || getp > maxp;  %check variables
            fprintf('-->Check (p). Selection is out of range.\n')
        elseif getp >= minp && getp <= maxp
            pvalid = true;
        end
    end
    %
    while nvalid == false   %assume invalid n value (relation to tablex)
        fprintf('How many observations (n)?\n')
        getn = input('--> ');   %the observations of interest (nprime)
        if getn < minn || getn > maxn;  %check variables
            fprintf('-->Check (n). Selection is out of range.\n')
        elseif getn >= minn && getn <= maxn
            nvalid = true;
        end
    end
    %
    if getp > getn;     %check n and p relation
        disp(['***There are more variables (p) than observations (n)' cr...
              'in this selection.  The lookup table is constrained  ' cr...
              'to no less than p = n.  Press ''ctrl''+''pause/break''' cr...
              'if you need to stop this script.***']); cr; cr;
        pvalid = false;     %give user a chance to reenter p
        nvalid = false;     %give user a chance to reenter n
    elseif getp <= getn     %&& pvalid == 1 && nvalid == 1
        paramvalid = true;  %good input parameters to the lookup table;
    end                     %exit input error checking
end
%
%************************************************************************
%**********************MATLAB variables usage*****************************
%---NOTE: All code originally written in this script.  Most of the variable
%referencing has been moved to functions that handle the scenarios listed
%in the box below this one.-----------------------------------------------
%'p' = "variable", 'n' = "observations", 'filename' = name of data file
%(not the name of the data matrix which is always X, by my default)
%'mev' = mean eigenvalue reference
%'getp' = variable we're going to find (user-supplied); consider p-prime
%'getn' = observations we're going to find (user-supplied); n-prime
%row = where rows of variables and observations are found in the data
%X = data matrix.  Lookup table reference tablex is a rename of X
%S = child of X-->the nearest neighbors of variables.  Used inside fx.
%Y = child of S-->the row entries of the nn observations on getn (in
%fx)
%'nn' --> "nearest neighbor" in all instances.
%     --> adding 'u' = "upper", 'l' = "lower", or 'p' and 'n' (see above)
%'ind' = "index" (of a row or column)
%************************************************************************
%There are seven scenarios that (p',n') that can be presented:
%1) p' not in table, n' not in table (both in range)-->interp p',n'
%2) p' not in table, n' in table (both in range)-->interp p', use n
%3) p' in table, n' not in table (both in range)-->use p, interp n'
```

```
%4) p' in table, n' in table (both in range)-->use p,n
%5) p' out of range, n' in range-->data below diagonal (illegal combo)
%6) p' in range, n' out of range-->largest recorded obs curve for interp p'
%7) p' out of range, n' out of range-->provide largest obs for largest p
%*************************************************************************
%******look for the nearest neighbor VARIABLES in the lookup table*********
[rp] = find(tablex(:,1)==getp);  %look for the input variable in the table;
[curves,nnup,nnlp,nnun,nnln] = findcurves(getp,getn,minp,maxp,minn,maxn);
%*********Call MCS on (p',n') as a proof-of-concept check***************
fprintf('Calling EigenMean for mean eigenvalues of (%d,%d)...',getp,getn)
[mevvec] = EigenMean(getp,getn,100);    %mean eigenvalue vector for (p',n')
                                        %mevvec is a matrix of row vectors
fprintf('Done!\n')
%
%plot the variables
fprintf('Plotting all curves...')
%create some new variables to increase graphing readability
minobs = nnln;
maxobs = nnun;
%set plot boundaries
xmin = 0.8;                 %left bound for x-axis
xmax = nnlp + 0.2;          %right bound for x-axis
ymin = 0;                   %lower bound for y-axis
ymax = curves(1,1);         %upper bound for y-axis; largest mean ev in data
%************************set plot vectors*********************************
%remaining variables have already been found; listed here for reference in
%terms of graphing ease. Order is the highest plot to the lowest plot
eind = size(curves,2);      %number of columns in the curve
curve1 = curves(1,:);       %nnun mev's for nnup
curve2 = curves(2,:);       %interpolated mev's for getn on nnun
curve3 = curves(3,:);       %nnun mev's for nnlp
curve4 = curves(4,:);       %interpolated mev's for getp
curve5 = mevvec(1:eind);    %from the MCS run
curve6 = curves(5,:);       %nnln mev's for nnlp
curve7 = curves(6,:);       %interpolated mev's for getn on nnln
curve8 = curves(7,:);       %nnun mev's for nnlp
xx = 1:eind;                %x-values; common to all plots
figure(1); box on; hold on;
axis([xmin xmax ymin ymax + 0.5]);
set(gca,'XTick',1:getp);    %display only integers on x axis
%
if getp > 30                %keep scaling under control
    set(gca,'XTick',floor(linspace(1,nnlp,10)))
end
%
plot(xx,curve1,'bs-','LineWidth',2)     %tabled mev's (nnup,nnln)
plot(xx,curve2,'b:')                    %upper interp model
plot(xx,curve3,'bo-','LineWidth',2)     %tabled mev's (nnup,nnun)
plot(xx,curve6,'ks-','LineWidth',2)     %tabled mev's (nnlp,nnln)
plot(xx,curve7,'k:')                    %lower interp model
plot(xx,curve8,'ko-','LineWidth',2)     %tabled mev's (nnlp,nnun)
plot(xx,curve4,'r:d')                   %Interpolated solution
plot(xx,curve5,'r','LineWidth',2)       %Actual eigenmean solution
%
%(A) goes here if needed.  See bottom of script.  All plot lines above.
line([xmin xmax],[1 1],'Color','k');    %Kaiser's criterion
%
```

```
%chart details
xlabel('Component(C_i)','FontSize',12)
ylabel('$$\mathsf{\bar \lambda}$$',...
    'interpreter','latex','fontsize',14)
title(['MEV Interpolation at Point (',...
        int2str(getp),',',int2str(getn),')'],'FontWeight','bold',...
        'FontSize',12)
%(B) goes here if needed.  See bottom of script.  Comment title above here.
%(C) goes here if needed.  See bottom of script.
%break  %uncomment if needed to run plain Horn's curve
legend(...
    ['Upper Nearest Neighbor  (',int2str(nnup),',',int2str(nnln),')'],...
    ['Upper Interpolated Curve (',int2str(nnup),',',int2str(getn),')'],...
    ['Upper Nearest Neighbor  (',int2str(nnup),',',int2str(nnun),')'],...
    ['Lower Nearest Neighbor  (',int2str(nnlp),',',int2str(nnln),')'],...
    ['Lower Interpolated Curve (',int2str(nnlp),',',int2str(getn),')'],...
    ['Lower Nearest Neighbor  (',int2str(nnlp),',',int2str(nnun),')'],...
    ['Interpolated Solution       (',int2str(getp),',',int2str(getn),')'],...
    ['Horn''s Algorithm Soln     (',int2str(getp),',',int2str(getn),')'],...
     'Location','NorthEast')
hold off
fprintf('Done!\n')
fprintf('***End of processing.***\n\n')
%
%end of program
%Extra stuff just to run a plain Horn's Curve
%(A)plot(xx,curve4,'r','LineWidth',2)       %Interpolated solution
%(B)title(['Random Data of Size (',...
%       int2str(getn),'x',int2str(getp),')'],'FontWeight','bold',...
%       'FontSize',12)
%(C)legend('Horn''s Curve','Kaiser''s Criterion (K1)','Location','NorthEast')
```

Main Script: HornsMethodRandom2OM.m

```
%Original code by Captain Andrew L. Bigley, USAF.  Written for partial
%fulfillment of a Master's of Science Degree in Operations Research, The
%Air Force Institute of Technology (AFIT), Wright-Patterson AFB, OH USA
%US Government disclaimer:
%The views expressed in this thesis are those of the author and do not
%reflect the official policy or position of the United States Air Force,
%Department of Defense, or the United States Government.
%This material is declared a work of the U.S. Government and is not subject
%to copyright protection in the United States.
%21 March 2013
%
%Graph the eigenvalue curves for one coordinate pair.  This script
%references a table of pre-determined, sorted mean eigenvalues and
%then interpolates to trap the (p',n') pair in the information in the
%table.  Various configurations that may be presented by a user to the
%script are discussed below.
%However, it differs from its ev_multiplots_MeanEV cousin in that it uses
%the 2OM coefficients and not the mean eigenvalues for direct reference***
%
%initialize the workspace
close all; clear all; clc
%initialize global variables--they are in the lookup table
global tablexbeta ssizep ssizen;    %these variables are global in nature

load LookupTableCoeffs.mat

maxp = max(tablexbeta(:,1));    %the largest variable value in the data
minp = min(tablexbeta(:,1));    %the smallest variable (a.o. 13 Jan minp=5)
                                %PCA on less than 5 variables?!?
maxn = max(tablexbeta(:,2));    %largest # of observations in the mapped data
minn = min(tablexbeta(:,2));    %smallest # of obs in the mapped data
cr = sprintf('\n');             %carriage return variable; use with 'disp'
%
%display opening message
disp(['This script will find Horn''s Curve as described by the    ' cr...
      'linear regression second-order model coefficients.         ' cr...
      'This version evaluates random data of selected (p'',n'').  ' cr...
      'For use with actual, real world datasets, see script       ' cr...
      'HornsMethodSampled2OM.']); cr;
disp(['The input values must be within these ranges:              ' cr...
      '                                                           ' ]);
fprintf('  # of variables (p)    --> {%d,%d} \n',minp,maxp)
fprintf('  # of observations (n) --> {%d,%d} \n',minn,maxn)
disp(['                                                           ' cr...
      'Also, make sure p is not less than n. ']); cr; cr;

%***check for input violations (out of range on p or n and n < p)***
%set datavalid flag to 'false'.  Assume the following:
paramvalid = false;     %valid input relation for p to n hasn't been rec'd
pvalid = false;         %valid variable input has not been entered
nvalid = false;         %valid observations input has not been entered
%
while paramvalid == false   %assume invalid parameters (relation of p,n)
    while pvalid == false   %assume invalid p value (relation to tablex)
```

```
        fprintf('Which variable (p) to graph?\n')
        getp = input('--> ');   %the variable of interest (pprime)
        if getp < minp || getp > maxp;  %check variables
            fprintf('-->Check (p). Selection is out of range.\n')
        elseif getp >= minp && getp <= maxp
            pvalid = true;
        end
    end
    %
    while nvalid == false   %assume invalid n value (relation to tablex)
        fprintf('How many observations (n)?\n')
        getn = input('--> ');   %the observations of interest (nprime)
        if getn < minn || getn > maxn;  %check variables
            fprintf('-->Check (n). Selection is out of range.\n')
        elseif getn >= minn && getn <= maxn
            nvalid = true;
        end
    end
    %
    if getp > getn;     %check n and p relation
        disp(['***There are more variables (p) than observations (n)' cr...
              'in this selection.  The lookup table is constrained  ' cr...
              'to no less than p = n.  Press ''ctrl''+''pause/break''' cr...
              'if you need to stop this script.***']); cr; cr;
        pvalid = false;     %give user a chance to reenter p
        nvalid = false;     %give user a chance to reenter n
    elseif getp <= getn %&& pvalid == 1 && nvalid == 1
        paramvalid = true;  %good input parameters to the lookup table;
    end                 %exit input error checking
    %
end
%
%*************************************************************************
%**********************MATLAB variables usage*****************************
%---NOTE: All code originally written in this script.  Most of the variable
%referencing has been moved to functions that handle the scenarios listed
%in the box below this one.-----------------------------------------------
%'p' = "variable", 'n' = "observations", 'filename' = name of data file
%(not the name of the data matrix which is always X, by my default)
%'mev' = mean eigenvalue reference
%'getp' = variable we're going to find (user-supplied); consider p-prime
%'getn' = observations we're going to find (user-supplied); n-prime
%row = where rows of variables and observations are found in the data
%X = data matrix.  Lookup table reference tablex is a rename of X
%S = child of X-->the nearest neighbors of variables.  Used inside fx.
%Y = child of S-->the row entries of the nn observations on getn (in
%fx)
%'nn' --> "nearest neighbor" in all instances.
%     --> adding 'u' = "upper", 'l' = "lower", or 'p' and 'n' (see above)
%'ind' = "index" (of a row or column)
%*************************************************************************
%There are seven scenarios that (p',n') that can be presented:
%1) p' not in table, n' not in table (both in range)-->interp p',n'
%2) p' not in table, n' in table (both in range)-->interp p', use n
%3) p' in table, n' not in table (both in range)-->use p, interp n'
%4) p' in table, n' in table (both in range)-->use p,n
%5) p' out of range, n' in range-->data below diagonal (illegal combo)
```

```
%6) p' in range, n' out of range-->largest recorded obs curve for interp p'
%7) p' out of range, n' out of range-->provide largest obs for largest p
%*************************************************************************
%******look for the nearest neighbor VARIABLES in the lookup table*********
[rp] = find(tablexbeta(:,1)==getp);  %look for the input variable in the
table;
%mevsolnone is the function "(m)ean (e)igen(v)alue (sol)utio(n) (one)"
%it does all the work for nearest neighbors in variables and observations
[curves,nnup,nnlp,nnun,nnln] = findcurves2OM(getp,getn,minp,maxp,minn,maxn);
%*********Call MCS on (p',n') as a proof-of-concept check**************
fprintf('Calling EigenMean for mean eigenvalues of (%d,%d)...',getp,getn)
[mevvec] = EigenMean(getp,getn,100);    %mean eigenvalue vector for (p',n')
                                        %mevvec is a matrix of row vectors
fprintf('Done!\n')
%
%plot the variables
fprintf('Plotting all curves...')
%set plot boundaries
xmin = 0.8;                 %left bound for x-axis
xmax = getp + 0.2;          %right bound for x-axis
ymin = 0;                   %lower bound for y-axis
ymax = curves(1,1);         %upper bound for y-axis; largest mean ev in data
%************************set plot vectors*********************************
%remaining variables have already been found; listed here for reference in
%terms of graphing ease. Order is the highest plot to the lowest plot
%plotting values are returned in curves matrix-->includes polyval, polyfit
eind = size(curves,2);      %number of columns in the curve
curve1 = curves(1,:);       %nnun mev's for nnup
curve2 = curves(2,:);       %interpolated mev's for getn on nnun
curve3 = curves(3,:);       %nnun mev's for nnlp
curve4 = curves(4,:);       %interpolated mev's for getp
curve5 = mevvec(1:eind);    %from the MCS run--yes, mean b/c random data
curve6 = curves(5,:);       %nnln mev's for nnlp
curve7 = curves(6,:);       %interpolated mev's for getn on nnln
curve8 = curves(7,:);       %nnun mev's for nnlp
xx = 1:eind;                %x-values; common to all plots
figure(1); box on; hold on;
axis([xmin xmax ymin ymax + 0.5]);
set(gca,'XTick',1:getp);    %display only integers on x axis
%
if getp > 30                %keep scaling under control
    set(gca,'XTick',floor(linspace(1,nnlp,10)))
end
%
plot(xx,curve1,'bs-','LineWidth',2)     %tabled mev's (nnup,nnln)
plot(xx,curve2,'b:')                    %upper interp model
plot(xx,curve3,'bo-','LineWidth',2)     %tabled mev's (nnup,nnun)
plot(xx,curve6,'ks-','LineWidth',2)     %tabled mev's (nnlp,nnln)
plot(xx,curve7,'k:')                    %lower interp model
plot(xx,curve8,'ko-','LineWidth',2)     %tabled mev's (nnlp,nnun)
plot(xx,curve4,'r:d')                   %Interpolated solution
plot(xx,curve5,'r','LineWidth',2)       %Actual eigenmean solution
%
line([xmin xmax],[1 1],'Color','k');    %Kaiser's criterion
%
%chart details
xlabel('Component (C_i)','FontSize',12)
```

```
ylabel('$$\mathsf{\hat \lambda}$$',...
    'interpreter','latex','fontsize',14)
title(['2OM Interpolation of Point (',int2str(getp),',',int2str(getn),...
       ')'],'FontWeight','bold','FontSize',12)
legend(...
    ['Upper Nearest Neighbor  (',int2str(nnup),',',int2str(nnln),')'],...
    ['Upper Interpolated Curve (',int2str(nnup),',',int2str(getn),')'],...
    ['Upper Nearest Neighbor  (',int2str(nnup),',',int2str(nnun),')'],...
    ['Lower Nearest Neighbor  (',int2str(nnlp),',',int2str(nnln),')'],...
    ['Lower Interpolated Curve (',int2str(nnlp),',',int2str(getn),')'],...
    ['Lower Nearest Neighbor  (',int2str(nnlp),',',int2str(nnun),')'],...
    ['Interpolated Solution       (',int2str(getp),',',int2str(getn),')'],...
    ['Horn''s Algorithm Soln     (',int2str(getp),',',int2str(getn),')'],...
     'Location','NorthEast')
hold off
fprintf('Done!\n')
fprintf('***End of processing.***\n\n')
%
%end of program
```

Main Script: HornsMethodSampledMEV.m

```
%Original code by Captain Andrew L. Bigley, USAF.  Written for partial
%fulfillment of a Master's of Science Degree in Operations Research, The
%Air Force Institute of Technology (AFIT), Wright-Patterson AFB, OH USA
%US Government disclaimer:
%The views expressed in this thesis are those of the author and do not
%reflect the official policy or position of the United States Air Force,
%Department of Defense, or the United States Government.
%This material is declared a work of the U.S. Government and is not subject
%to copyright protection in the United States.
%21 March 2013
%
%Graph the eigenvalue curves for a sampled (real-world) dataset.
%This script references a table of pre-determined, sorted mean eigenvalues
%and then interpolates to trap the (p',n') pair in the information in the
%table.  Various configurations that may be presented by a user to the
%script are discussed below.
%
%initialize the workspace
close all; clear all; clc
%initialize global variables--they are in the lookup table
global tablex ssizep ssizen;    %these variables are global in nature
%
%load the lookup table.  Do not confuse with a sample dataset!!!
%Format expected by the program is [p n meaneigenvalues] where
%-->p = # variables => sorted descending;
%-->n = # observations => sorted ascending;
%-->meaneigenvalues = data elements => sorted descending from EigenMean
load LookupTable.mat
%
%initialize local variables
validdata = false;          %flag to stay in the input loop
smallsamp = false;          %data is 2-4 variables & can be direct calc'd
maxp = max(tablex(:,1));    %the largest variable value in the data
minp = min(tablex(:,1));    %the smallest variable (a.o. 13 Jan minp=5)
                            %PCA on less than 5 variables?!?
maxn = max(tablex(:,2));    %largest # of observations in the mapped data
minn = min(tablex(:,2));    %smallest # of obs in the mapped data
cr = sprintf('\n');         %carriage return variable; use with 'disp'
%
%display opening message
disp(['This script will find Horn''s Curve to aid in making a       ' cr...
      'Principal Components Analysis (PCA) dimensionality deter-    ' cr...
      'mination for an actual--sampled--data set.  Horn''s Curve    ' cr...
      'is found by interpolating known, "ideal" data of size        ' cr...
      'equivalent to the actual sample size.  Constraints regard-   ' cr...
      'ing input and what the script can do are listed below.      ']); cr;
disp(['The input values must be within these ranges:                ' cr...
      '                                                            ']); cr;
fprintf('  # of variables (p)    --> {%d,%d} \n',minp,maxp)
fprintf('  # of observations (n) --> {%d,%d} \n',minn,maxn)
disp(['                                                             ' cr...
      'A crucial condition to consider is underdetermined data; that' cr...
      'is, data having fewer observations n than features p.  PCA of' cr...
      'underdetermined data is possible; however, this script does  ' cr...
```

```
      'not accept such datasets.                                          ' cr...
      '                                                                   ' cr...
      'Please choose a dataset to load.  Type the number and press  ' cr...
      '''Enter.''  If the dataset is not listed, choose ''0'' (zero)' cr...
      'and type in the filename.                                          ' cr...
      '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~'
cr...
      '(1)  Forest Fires                                                  ' cr...
      '(2)  Glass                                                         ' cr...
      '(3)  Parkinsons                                                    ' cr...
      '(4)  SECOM                                                         ' cr...
      '(5)  Seeds                                                         ' cr...
      '(6)  Semeion Handwriting Characters                                ' cr...
      '(7)  Steel Plates                                                  ' cr...
      '(8)  Wisconsin Breast Cancer Study                                 ' cr...
      '(9)  Wines (Set 1)                                                 ' cr...
      '(10) Wines (Set 2)                                                 ' cr...
      '.....................................................................'
cr...
      '(0)  Manually enter a filename                                     ' cr...

'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~'']);cr;
while validdata == false;
    sel = input('-> Please make a selection (1-10) or (0): ');
    %find out which dataset to load based upon user's selection
    switch sel
        case 1
            filename = 'ForestFires';
        case 2
            filename = 'Glass';
        case 3
            filename = 'Parkinsons';
        case 4
            filename = 'SECOM';
        case 5
            filename = 'Seeds';
        case 6
            filename = 'Semeion';
        case 7
            filename = 'SteelPlates';
        case 8
            filename = 'WIBreastCancer';
        case 9
            filename = 'Wines1';
        case 10
            filename = 'Wines2';
        otherwise
            fprintf('Please enter the filename (script assumes .mat)\n')
            filename = input('--> ','s');
    end
    load(filename,'-mat');  %the datafile to load
    [getn,getp] = size(X);  %size of the data
    %check for all the conditions of X (min size, max size, n>p)
    if getn >= minn && getn <= maxn && getp >= minp && getp <= maxp
        validdata = true;   %binary that data is valid for processing
    elseif getp > 1 && getp < 5
        validdata = true;   %binary that data is valid for processing
```

```
        smallsamp = true;   %binary that direct computation is valid
        fprintf('The selected filename has data small enough for a direct\n')
        fprintf('calculation of Horn''s curve.\n\n')
    else
        fprintf('The sample data is size %d x %d (obs x var).\n',getn,getp)
        fprintf('The lookup table is (%d to %d) observations and
\n',minn,maxn)
        fprintf('(%d to %d) variables.  Please make another
selection.\n\n',minp,maxp)
    end

    if getp > getn;     %check n and p relation--no underdetermined sets!
        disp(['***There are more variables (p) than observations (n)' cr...
            'in this data.  The lookup table is constrained to no ' cr...
            'less than p = n.  Press ''ctrl''+''pause/break''' cr...
            'to stop this script.***']); cr; cr;
        validdata = false;
    end
end
%
%call the interpolating function for p >= 5
%*************************************************************************
if smallsamp == false;              %skip interpolation for small data
    [curves,nnup,nnlp,nnun,nnln]=findcurves(getp,getn,minp,maxp,minn,maxn);
else
    ssev = EigenMean(getn,getp,100);%small sample eigenvalues
    curves = zeros(7,getp);
    nnlp = getp;
    nnup = getp;
    nnln = getn;
    nnun = getn;
end
%************************Get sorted eigenvalues of X*********************
fprintf('Getting eigenvalues of (%dx%d)...',getn,getp)
R = corr(X);                        %X is the data matrix
[V,D] = eig(R);                     %Don't need V eigenvectors (use for loads);
                                    %do need D eigenvalues
sev = sort(diag(D),'descend')'; %'sev'=sampled data eigenvalues
fprintf('Done!\n')
%*************************************************************************
%plot the variables
fprintf('Plotting all curves...')
%create some new variables to increase graphing readability
eind = size(curves,2);  %number of columns in the curve
esev = size(sev,2);     %want to show all the eigenvalues
%set plot boundaries
xmin = 0.8;             %left bound for x-axis
xmax = esev + 0.2;      %right bound for x-axis
ymin = 0;               %lower bound for y-axis
ymax = sev(1) + 0.5;    %upper bound for y-axis; largest ev in data
%find out where the two lines (interpolated and actual) cross; this is the
%entire utility of Horn's Curve in a nutshell
%************************set plot vectors*********************************
%remaining variables have already been found; listed here for reference in
%terms of graphing ease. Order is the highest plot to the lowest plot
%what's the case with Horn's curve?  Small dataset = special situation.
%Note: 'curves' gets returned with all nearest neighbor & surrogate curves
```

```
if smallsamp == false
    curve4=curves(4,:); %interp sev's for getp
else
    curve4=ssev;        %small sample eigenvalues
end
xx = 1:eind;            %x-values; common to all interpolated curves
screeX = 1:esev;        %vector of indices in sampled data eigenvalue
screeY = sev;           %sampled data eigenvalues
d1 = length(curve4);    %length of curve4
screeRX = zeros(1,d1);  %preallocate the vector.
screeRY = zeros(1,d1);  %eigenvalues that would be discarded by Horn's Test
count = 1;              %reset counter in outer loop
qcount = 1;             %reset counter for inner loop
qflag = false;          %was a contested (for dimen.) eigenvalue found?
%get the eigenvalues that Horn's test indicates are less than the dim.
%they will be colored gray in the plot
for i=1:d1
    if (curve4(i) > sev(i));    %ck for point above Horn's curve
        screeRX(count) = i;
        screeRY(count) = sev(i);
        count = count + 1;
        %check to see if a point less than Horn's curve is > 1.  These are
        %eigenvalues that we would keep just by using Kaiser's criterion
        if sev(i) > 1
            qflag = true;
            qx(1,qcount) = i;
            qy(1,qcount) = sev(i);
            qcount = qcount + 1;
            %next 8 lines summarize for the screen.  Will only show if this
sub
            %is run.
            pctvar2 = sum((qy)/getp)*100;
            cpc = length(qx);               %# of contested prin components
        end
    end
end
%clear up extra zeros in screeRX, screeRY
%if the input variable getp is a tabled value, there is nothing left to add
%and the remaining zeros in the array should be truncated.
%otherwise (at the 'else') reindex the array and count-in the smallest
%eigenvalues of sev that extend beyond the length of curve4.  The smallest
%eigenvalues will be plotted but colored so that it's obvious they are not
%considered significant to PCA.
if nnlp == nnup             %getp is a direct match in the table
    [rt] = find(screeRX == 0);
    screeRX(rt) = [];       %truncate trailing zeros
    screeRY(rt) = [];       %truncate trailing zeros
else
    d2 = screeRX(1):getp;
    screeRX = d2;           %need the integers from the first cutoff eigen-
    screeRY = sev(d2);      %value to the total number in sev
end
%
%Capture the PC and pct variance for Horn's
pchorns = length(screeX)-length(screeRX);   %find the #ev's above the curve
pctvar1 = sum((screeY(1:pchorns))/getp)*100;
%
```

```
figure(1); box on; hold on;
set(gca,'XTick',1:getp);    %display only integers on x axis
%
if getp > 30                %keep scaling under control
    set(gca,'XTick',floor(linspace(1,nnlp,10)))
    xmin = 0;             %large values--give some more whitespace
    xmax = getp+getp*0.02; %
end
axis([xmin xmax ymin ymax]);
%
%uncomment these boxes to see where the tabled & interpolated curves are
%doing so will royally mess up the legend entries--therefore not
%recommended for 'long term' use
%plot(xx,curve1,'bs-','LineWidth',2)    %tabled sev's (nnup,nnln)
%plot(xx,curve2,'b:')                   %upper interp model
%plot(xx,curve3,'bo-','LineWidth',2)    %tabled sev's (nnup,nnun)
%plot(xx,curve6,'rs-','LineWidth',2)    %tabled sev's (nnlp,nnln)
%plot(xx,curve7,'r:')                   %lower interp model
%plot(xx,curve8,'ro-','LineWidth',2)    %tabled sev's (nnlp,nnun)
%plot(screeX,screeY,'b','LineWidth',2); %sampled data scree line (A)
plot(screeX,screeY,'b','LineWidth',2);  %sampled data scree line
plot(xx,curve4,'r','LineWidth',2)       %Interpolated Horn's Curve soln
line([xmin xmax],[1 1],'Color','k');    %Kaiser's criterion
%scatter(screeX,screeY,64,[1 0.5 0.3],'filled','MarkerEdgeColor','k');(B)
%this line above has good scatterplot color; looking for green, though
scatter(screeX,screeY,64,[0 0.9 0.2],'filled','MarkerEdgeColor','k');
scatter(screeRX,screeRY,64,[0.9 0.9 0.9],'filled','MarkerEdgeColor','k');
if qflag == true   %contested eigenvalues betw. Horn's & K1.  Display in red
    scatter(qx,qy,64,[1 0 0],'filled','MarkerEdgeColor','k');
end
%
%chart details
xlabel('Component (C_i)','FontSize',12)
ylabel('Eigenvalue ( \lambda)','FontSize',12)
%ylabel('\lambda','FontSize',12)
legend('Scree Line','Horn''s Curve','Kaiser''s Criterion (K1)',...
       'Location','NorthEast')
%title('Sampled Data of Size (517x13)','FontSize',12,'FontWeight','bold')
title(['sev Interpolated Solution of Dataset "',filename,...
       '" (',int2str(getn),'x',int2str(getp),')'],...
       'FontWeight','bold','FontSize',12)
%break;          %(D)uncomment if running plain curve
if qflag == true;   %choose the correct legend; did we contest components?
                    %if qflag = 1, then yes there is something here
    legend('Scree Line','Horn''s Curve',...
        'Kaiser''s Criterion (K1) = 1.0',...
       ['\bf',int2str(pchorns),' \rm \lambda > Horn''s Curve > K1'],...
       ['\bf',int2str(getp-pchorns-cpc),...
        ' \rm \lambda < Horn''s Curve < K1'],...
       ['\bf',int2str(cpc),' \rm K1 < \lambda < Horn''s Curve'],...
       'Location','NorthEast')
elseif qflag == false;  %no contested points--both tests are in agreement
    legend('Scree Line','Horn''s Curve',...
           'Kaiser''s Criterion (K1) = 1.0',...
          ['\bf',int2str(pchorns),...
           ' \rm \lambda > Horn''s Curve > K1'],...
          ['\bf',int2str(getp-pchorns),...
```

```
           ' \rm \lambda < Horn''s Curve < K1'],...
       'Location','NorthEast')
end
hold off
fprintf('Done!\n')
%
%send summary to the screen
disp('************************** Summary ****************************');
cr;
if qcount > 1;  %qcount = #of contested eigenvalues for dimensionality
                %because qcount is also a loop marker, #ev's = val -1
    fprintf('Dimensionality is estimated at %d principal components
by\n',pchorns)
    fprintf('Horn''s test.  There are a total of %d eigenvalues below
Horn''s\n',cpc)
    disp(['curve and greater than K1.  These eigenvalues should be further'
cr...
          'evaluated against additional criteria for usefulness.' cr...
          '(Additional criteria == qualitative and quantitative   ' cr...
          'aspects of the study that are particular to a dataset, ' cr...
          'purpose of the study, and analyst selection.)  ']);cr;
    fprintf('\n')
    fprintf('-->Component #: %d  Eigenvalue: %2.4f <--\n',[qx; qy]);
    fprintf('\n')
    fprintf('Proportion of total variance explained by Horn''s =
%2.2f%%.\n',pctvar1);
    fprintf('Additional proportion of total variance explained by the\n')
    fprintf('contested "between the curves" components: %2.2f%%.\n',pctvar2)
    fprintf('If %d contested components are included, proportion =
%2.2f%%.\n',cpc,pctvar1+pctvar2)
else
    fprintf('Dimensionality is estimated at %d principal
components.\n',pchorns)
    fprintf('Proportion of total variance explained = %2.2f%%.\n',pctvar1)
    fprintf('There are no contested components.\n')
end
disp('***************************************************************');
fprintf('THE FILENAME USED IN THIS ANALYSIS IS %s  \n',filename)
fprintf('\nEnd of processing.\n\n')
%
%end of script
```

Main Script: HornsMethodSampled2OM.m

```
%Original code by Captain Andrew L. Bigley, USAF.  Written for partial
%fulfillment of a Master's of Science Degree in Operations Research, The
%Air Force Institute of Technology (AFIT), Wright-Patterson AFB, OH USA
%US Government disclaimer:
%The views expressed in this thesis are those of the author and do not
%reflect the official policy or position of the United States Air Force,
%Department of Defense, or the United States Government.
%This material is declared a work of the U.S. Government and is not subject
%to copyright protection in the United States.
%21 March 2013
%
%Graph the eigenvalue curves for a sampled (real-world) dataset.
%This script references a table of pre-determined, sorted mean eigenvalues
%and then interpolates to trap the (p',n') pair in the information in the
%table.  Various configurations that may be presented by a user to the
%script are discussed below.
%
%initialize the workspace
close all; clear all; clc
%initialize global variables--they are in the lookup table
global tablexbeta ssizep ssizen;    %these variables are global in nature
%
%load the lookup table.  Do not confuse with a sample dataset X!!!
%Format expected by the program is tablexbeta = [p n b2 b1 b0] where
%-->p = # variables => sorted descending;
%-->n = # observations => sorted ascending;
%-->{b2 b1 b0} = data elements => coefficients from the lookup table
load LookupTableCoeffs.mat
%
%initialize local variables
validdata = false;              %boolean flag to stay in the input loop
smallsamp = false;              %data is 2-4 variables & can be direct calc'd
maxp = max(tablexbeta(:,1));    %the largest variable value in the data
minp = min(tablexbeta(:,1));    %the smallest variable (a.o. 13 Jan minp=5)
                                %PCA on less than 5 variables?!?
maxn = max(tablexbeta(:,2));    %largest # of observations in the mapped data
minn = min(tablexbeta(:,2));    %smallest # of obs in the mapped data
cr = sprintf('\n');             %carriage return variable; use with 'disp'
%
%display opening message
disp(['This script will find Horn''s Curve to aid in making a       ' cr...
      'Principal Components Analysis (PCA) dimensionality deter-    ' cr...
      'mination for an actual--sampled--data set.  Horn''s Curve    ' cr...
      'is found by interpolating known, "ideal" data of size        ' cr...
      'equivalent to the actual sample size.  Constraints regard-   ' cr...
      'ing input and what the script can do are listed below.      ']); cr;
disp(['The input values must be within these ranges:                ' cr...
      '                                                            ']); cr;
fprintf('  # of variables (p)    --> {%d,%d} \n',minp,maxp)
fprintf('  # of observations (n) --> {%d,%d} \n',minn,maxn)
disp(['                                                             ' cr...
      'A crucial condition to consider is underdetermined data; that' cr...
      'is, data having fewer instances n than features p.  PCA of   ' cr...
      'underdetermined data is possible; however, this script does  ' cr...
```

```
      'not accept such datasets.                                          ' cr...
      '                                                                   ' cr...
      'Please choose a dataset to load.  Type the number and press  ' cr...
      '''Enter.''  If the dataset is not listed, choose ''0'' (zero)' cr...
      'and type in the filename.                                          ' cr...
      '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~'
cr...
      '(1)  Forest Fires                                                  ' cr...
      '(2)  Glass                                                         ' cr...
      '(3)  Parkinsons                                                    ' cr...
      '(4)  SECOM                                                         ' cr...
      '(5)  Seeds                                                         ' cr...
      '(6)  Semeion Handwriting Characters                                ' cr...
      '(7)  Steel Plates                                                  ' cr...
      '(8)  Wisconsin Breast Cancer Study                                 ' cr...
      '(9)  Wines (Set 1)                                                 ' cr...
      '(10) Wines (Set 2)                                                 ' cr...
      '.............................................................      '
cr...
      '(0)  Manually enter a filename                                     ' cr...

'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~'']);cr;
while validdata == false;
    sel = input('-> Please make a selection (1-10) or (0): ');
    %find out which dataset to load based upon user's selection
    switch sel
        case 1
            filename = 'ForestFires';
        case 2
            filename = 'Glass';
        case 3
            filename = 'Parkinsons';
        case 4
            filename = 'SECOM';
        case 5
            filename = 'Seeds';
        case 6
            filename = 'Semeion';
        case 7
            filename = 'SteelPlates';
        case 8
            filename = 'WIBreastCancer';
        case 9
            filename = 'Wines1';
        case 10
            filename = 'Wines2';
        otherwise
            fprintf('Please enter the filename (script assumes .mat)\n')
            filename = input('--> ','s');
    end
    load(filename,'-mat');  %the datafile to load
    [getn,getp] = size(X);  %size of the data
    %check for all the conditions of X (min size, max size, n>p)
    if getn >= minn && getn <= maxn && getp >= minp && getp <= maxp
        validdata = true;
    elseif getp > 1 && getp < 5
        validdata = true;   %binary that data is valid for processing
```

```
        smallsamp = true;    %binary that direct computation is valid
        fprintf('The selected filename has data small enough for a direct\n')
        fprintf('calculation of Horn''s curve.\n\n')
    else
        fprintf('The sample data is size %d x %d (obs x var).\n',getn,getp)
        fprintf('The lookup table is (%d to %d) observations and
\n',minn,maxn)
        fprintf('(%d to %d) variables.  Please make another
selection.\n\n',minp,maxp)
    end

    if getp > getn;      %check n and p relation
        disp(['***There are more variables (p) than observations (n)' cr...
            'in this data.  The lookup table is constrained to no ' cr...
            'less than p = n.  Press ''ctrl''+''pause/break''' cr...
            'to stop this script.***']); cr; cr;
        validdata = false;
    end
end
%
%call the interpolating function
%*************************************************************************
if smallsamp == false;               %skip interpolation for small data
  [curves,nnup,nnlp,nnun,nnln]=findcurves2OM(getp,getn,minp,maxp,minn,maxn);
else
   ssev = EigenMean(getn,getp,100);  %small sample eigenvalues
   curves = zeros(7,getp);
   nnlp = getp;
   nnup = getp;
   nnln = getn;
   nnun = getn;
end
%************************Get sorted eigenvalues of X*********************
fprintf('Getting eigenvalues of (%dx%d) sampled data...',getn,getp)
R = corr(X);                        %X is the data matrix
[V,D] = eig(R);                     %Don't need Vy eigenvectors;
                                    %do need Dy eigenvalues
screeY = sort(diag(D),'descend')';
fprintf('Done!\n')
%*************************************************************************
%plot the variables
fprintf('Plotting all curves...')
%create some new variables to increase graphing readability
cols = size(curves,2);      %number of columns in the curve
screeX = 1:cols;            %vector of indices in sampled data eigenvalue
%set plot boundaries
xmin = 0.8;                 %left bound for x-axis
xmax = cols + 0.2;          %right bound for x-axis
ymin = 0;                   %lower bound for y-axis
ymax = screeY(1) + 0.5;     %upper bound for y-axis; largest ev in data
%find out where the two lines (interpolated and actual) cross; this is the
%entire utility of Horn's Curve in a nutshell
%************************set plot vectors*********************************
%Note: 'curves' returns all NN and surrogate curves; only 'curve4' needed
%what's the case with Horn's curve?  Small dataset = special situation
if smallsamp == false
    curve4=curves(4,:);     %interp betas for getp
```

```
else
    curve4=ssev;            %small sample eigenvalues
end
d1 = length(curve4);        %length of curve4--the est. Horn's curve
screeRX = zeros(1,d1);      %preallocate the vector.
screeRY = zeros(1,d1);      %eigenvalues that would be discarded by Horn's
count = 1;                  %reset counter in outer loop
qcount = 1;                 %reset counter for inner loop
qflag = 0;                  %was a contested (for dimen.) eigenvalue found?
for i=1:d1
    if curve4(i) > screeY(i)   %screeY is the curve of sampled eigenvalues
        screeRX(count) = i;
        screeRY(count) = screeY(i);
        count = count + 1;
        %check to see if a point less than Horn's curve is > 1.  These are
        %eigenvalues that we would keep just by using Kaiser's criterion
        if screeY(i) > 1
            qflag = 1;
            qx(1,qcount) = i;
            qy(1,qcount) = screeY(i);
            qcount = qcount + 1;
            pctvar2 = sum((qy)/getp)*100;
            cpc = length(qx);   %# of contested prin components
        end
    end
end

d2 = screeRX(1):getp;
screeRX = d2;               %need the integers from the first cutoff eigen-
screeRY = screeY(d2);       %value to the total number in mev
%so how did the PC dimensionality estimate go?  Capture the PC for Horn's
pchorns = length(screeX)-length(screeRX);   %find the #ev's above the curve
pctvar1 = sum((screeY(1:pchorns))/getp)*100;
%.....................................................................
figure(1); box on; hold on;
set(gca,'XTick',1:getp);    %display only integers on x axis
%
if getp > 30                %keep scaling under control
    set(gca,'XTick',floor(linspace(1,nnlp,10)))
    xmin = 0;               %large values--give some more whitespace
    xmax = getp+getp*0.02;  %
end
axis([xmin xmax ymin ymax]);
plot(screeX,screeY,'b','LineWidth',2)   %Scree line
plot(screeX,curve4,'r','LineWidth',2)   %Interpolated Horn's Curve soln
line([xmin xmax],[1 1],'Color','k')     %Kaiser's criterion
scatter(screeX,screeY,64,[0 0.9 0.2],'filled','MarkerEdgeColor','k');
scatter(screeRX,screeRY,64,[0.9 0.9 0.9],'filled','MarkerEdgeColor','k');
if qflag == true;   %contested eigenvalues betw. Horn's & K1.  Display in red
    scatter(qx,qy,64,[1 0 0],'filled','MarkerEdgeColor','k');
end
%chart details
xlabel('Component (C_i)','FontSize',12)
ylabel('Eigenvalue ( \lambda)','FontSize',12)
title(['2OM Interpolated Solution of Dataset "',filename,...
       '" (',int2str(getn),'x',int2str(getp),')'],...
       'FontWeight','bold','FontSize',12)
```

```
if qflag == true;   %choose the correct legend; did we contest components?
                    %if true, then yes there is something here
    legend('Scree Line','Horn''s Curve',...
           'Kaiser''s Criterion (K1) = 1.0',...
          ['\bf',int2str(pchorns),...
           ' \rm \lambda > Horn''s Curve > K1'],...
          ['\bf',int2str(getp-pchorns-cpc),...
           ' \rm \lambda < Horn''s Curve < K1'],...
          ['\bf',int2str(cpc),...
           ' \rm K1 < \lambda < Horn''s Curve'],...
       'Location','NorthEast')
elseif qflag == false;  %no contested points--both tests are in agreement
    legend('Scree Line','Horn''s Curve',...
           'Kaiser''s Criterion (K1) = 1.0',...
          ['\bf',int2str(pchorns),...
           ' \rm \lambda > Horn''s Curve > K1'],...
          ['\bf',int2str(getp-pchorns),...
           ' \rm \lambda < Horn''s Curve < K1'],...
           'Location','NorthEast')
end
hold off
fprintf('Done!\n')
%send summary to the screen
disp('************************** Summary ****************************');
cr;
if qcount > 1;  %qcount = #of contested eigenvalues for dimensionality
                %because qcount is also a loop marker, #ev's = val -1
    fprintf('Dimensionality is estimated at %d principal components
by\n',pchorns)
    fprintf('Horn''s test.  There are a total of %d eigenvalues below
Horn''s\n',cpc)
    disp(['curve and greater than K1.  These eigenvalues should be further'
cr...
          'evaluated against additional criteria for usefulness.' cr...
          '(Additional criteria == qualitative and quantitative   ' cr...
          'aspects of the study that are particular to a dataset, ' cr...
          'purpose of the study, and analyst selection.)  ']);cr;
    fprintf('\n')
    fprintf('-->Component #: %d  Eigenvalue: %2.4f <--\n',[qx; qy]);
    fprintf('\n')
    fprintf('Proportion of total variance explained by Horn''s =
%2.2f%%.\n',pctvar1);
    fprintf('Additional proportion of total variance explained by the\n')
    fprintf('contested "between the curves" components: %2.2f%%.\n',pctvar2)
    fprintf('If %d contested components are included, proportion =
%2.2f%%.\n',cpc,pctvar1+pctvar2)
else
    fprintf('Dimensionality is estimated at %d principal
components.\n',pchorns)
    fprintf('Proportion of total variance explained = %2.2f%%.\n',pctvar1)
    fprintf('There are no contested components.\n')
end
disp('***************************************************************');
fprintf('THE FILENAME USED IN THIS ANALYSIS IS %s  \n',filename)
fprintf('\nEnd of processing.\n\n')
%
%end of script
```

Supporting Function: EigenMean.m

```
function[meanev] = EigenMean(p,n,k)
%Original code by Captain Andrew L. Bigley, USAF.  Written for partial
%fulfillment of a Master's of Science Degree in Operations Research, The
%Air Force Institute of Technology (AFIT), Wright-Patterson AFB, OH USA
%US Government disclaimer:
%The views expressed in this thesis are those of the author and do not
%reflect the official policy or position of the United States Air Force,
%Department of Defense, or the United States Government.
%This material is declared a work of the U.S. Government and is not subject
%to copyright protection in the United States.
%21 March 2013
%NOTE:  The comment lines above can be removed w/no loss of function help.
%This function finds the mean eigenvalues for a n x p data set.  The mean
%eigenvalues are useful for performing Principal Components Analysis and
%are found through repeated iterations of normal random probability distri-
%bution calls and subsequent determination of the correlation matrix of
%size n x p.  The random number draws are done in Monte Carlo simulation
%iterations of size k.  The function structure is:
%
%[meanev] = EigenMean(p,n,k)
%
%Inputs: p = # of variables;
%        n = # of observations;
%        k = # of Monte Carlo simulation iterations.
%
%Outputs: [meanev] is a return vector of size (1 x p) containing the
%sorted and averaged eigenvalues by index (#eigenvalues = #variables).
%All statistical assumptions are based upon data being distributed standard
%normal (mean = 0, standard deviation = 1).  The correlation operator is
%applied to the random data matrix before extracting the eigenvalues.
%
%**The function will warn when:
%*1) p is overfitted to n; that is, n should be at least as large as p and
%preferably 3p <= n.  Note that large values of k (>1000) will result in
%long processing times, and in instances with large (p,n) combinations it
%will appear that MATLAB has stopped responding. Therefore, unless utmost
%precision in the mean eigenvalues result is needed, such as least-squares
%model fitting where precise coefficient estimates must be made, consider
%using k = 1000 (the default setting if k is not provided).  For curve
%fitting using interpolation methods, k = 100 may prove satisfactory if p
%is sufficiently large to "spread" the variation among more eigenvalues.
%The tradeoff is more iterations push towards convergence of the true means
%at the expense of processing time (minutes are not uncommon if p,n,k are
%as small as a 500 each).  Longer times (hours) are not out the question if
%the input parameters are in the thousands.
%*2) If a warning regarding eigenvalue computations is received due to
%non-real or singular results, check for lack of linear independence in the
%input matrix columns.  One or more variables are dependent on another.
%
%(Function version a.o. 14 Jan 13)

error(nargchk(2,3,nargin))
if nargin == 2              %need at least two inputs (p,n)
    fprintf('(Using default MCS iterations k=100)\n')
```

```
    k = 100;
end

if p > n
    fprintf('Data is overfitted (p>n).  Check your input.\n')
    fprintf('Computations will complete but the smallest eigenvalues\n ')
    fprintf('reduce to approximately zero.\n')
end
%get started; initialize variables
mev = diag(zeros(p));               %size of eigenvalue mean accumulator
                                    %variable
for i=1:k
    Q = normrnd(0,1,n,p);           %Generate Y--a random ~Norm(0,1)
                                    %matrix of n x p size
    Ry = corr(Q);                   %Correlation matrix of X
    [Vy,Dy] = eig(Ry);              %Get (Vy) eigenvectors and
                                    %(Dy) eigenvalues for correlation of Y
    Dy = sort(diag(Dy),'descend');  %too big for 'eigs'; sort 'eig' result
    mev = mev + Dy;                 %adds each eigenvalue by array index
end
%
meanev = (mev.*(1/k))';             %return vector is the mean eigenvalues
                                    %of the correlation matrix sorted by
                                    %index
%
%end of function
```

Supporting Function: findcurves.m

```
function [curves,nnup,nnlp,nnun,nnln] = findcurves(p,n,minp,maxp,minn,maxn)
%Original code by Captain Andrew L. Bigley, USAF.  Written for partial
%fulfillment of a Master's of Science Degree in Operations Research, The
%Air Force Institute of Technology (AFIT), Wright-Patterson AFB, OH USA
%US Government disclaimer:
%The views expressed in this thesis are those of the author and do not
%reflect the official policy or position of the United States Air Force,
%Department of Defense, or the United States Government.
%This material is declared a work of the U.S. Government and is not subject
%to copyright protection in the United States.
%21 March 2013
%
%Mean Eigenvalue Solution, Case I:
% * Both the variable (p') and observation (n') is in the range of the    *
% * lookup table mapped values.  Also, (p') and (n') are not the minimum  *
% * nor maximum values in the table and the nearest neighbors method will *
% * achieve satisfactory results.                                          *
%
%The function returns six curves to the main executable:
%1) Upper nearest neighbor variable, lower nearest neighbor observation.
%2) Interpolated curve for p', based upon upper p & lower/upper obs.
%3) Upper nearest neighbor variable, upper nearest neighbor observation.
%4) Lower nearest neighbor variable, lower nearest neighbor observation.
%6) Interpolated curve for p', based upon lower p & lower/upper obs.
%7) Lower nearest neighbor variable, upper nearest neighbor observation.
%
global tablex ssizep ssizen
%
rnnup = []; %init the data row variable; also acts as a flag to search
rnnlp = []; %init the data row variable; also acts as a flag to search
rnnun = []; %init the data row variable; also acts as a flag to search
rnnln = []; %init the data row variable; also acts as a flag to search
%*************************************************************************
%******Check to see if data is already in the lookup table and at what*****
%*extremes.  Data at or near an edge will need to be conditioned to accept*
%*something else than what the nearest neighbor search algorithm assigns***
%******Variable assignment*************************************************
[rp] = find(tablex(:,1) == p);  %look for the input variable
if isempty(rp) == false;        %found p' in tablex but where is it?
    %p' is the minimum variable
    if p == minp;               %does p' = min table variable value?
        nnlp = p;               %yes, assign lower neighbor to it
        nnup = minp + ssizep;   %upper nearest neighbor is a stepsize up
        rnnup = rp;             %abort the nnup, nnlp searches because
        rnnlp = rp;             %variable assignments are made
    %p' is the maximum variable
    elseif p == maxp;
        nnlp = maxp - ssizep;   %go a stepsize down for nnlp
        nnup = p;               %already at highest variable value
        rnnup = rp;             %abort the nnup, nnlp searches because
        rnnlp = rp;             %variable assignments are made
    %p' is somewhere in the middle; in this case, pass all known info fwd.
    elseif p > minp && p < maxp
        nnlp = p;
```

```
        nnup = p;
        rnnup = rp;
        rnnlp = rp;
    end
end
%
if isempty(rp) == true; %did not find p' in the lookup table; search!
    ind = 0;                                    %reset the search index
    while isempty(rnnup) && isempty(rp)         %run loop while empty
        ind = ind+1;                            %incr the array counter
        [rnnup] = find(tablex(:,1) == p+ind);   %add the index
        if ind > ssizep;    %lookup table is corrupted
            fprintf('WARNING: Variable stepsize exceeded. Check tablex.\n')
            rp = -1;        %value indicates we had a problem here.
            break;          %let program critical stop
        end
    end
    nnup = p+ind;                               %got the upper neighbor
    %now find the lower nearest neighbor variable
    ind = 0;                                    %reset the search index
    while isempty(rnnlp) && isempty(rp)         %run loop while empty
        ind = ind+1;                            %incr the array counter
        [rnnlp] = find(tablex(:,1) == p-ind);   %subtract the index
        if ind > ssizep;    %lookup table is corrupted
            fprintf('WARNING: Variable stepsize exceeded. Check tablex.\n')
            rp = -2;        %value indicates we had a problem here.
            break;          %let program critical stop
        end
    end
    nnlp = p-ind;           %got the lower neighbor
end             %exit out of looking for the variable neighbor indices
%
%truncate the data into a subset of tablex meaningful to the analysis
%create sub-matrix S of only the rows of p(-) and p(+)
S = [tablex(rnnup,:); tablex(rnnlp,:)];
trimp = find(S(end,:) > 0);  %inspect the last row because it's p(-)
S = S(:,trimp); %eliminate sparsity in S; upper/lower vars equal length
%S is the reduced matrix to work from for observation nearest neighbors****
%***********************Done with VARIABLES*******************************
%***find the OBSERVATION nearest neighbors in the truncated data matrix S**
%find the start of zero entries in the lower bound (nnlp) and then trim
%each bound (upper and lower) to that length.  Purpose: set equal number of
%variables in the vectors of matrix S
[rn] = find(S(:,2) == n);       %look for the input observation
if isempty(rn) == 0;            %found n in S but where is it?
    %n is the minimum observation
    if n == minn;               %is the input = min(S) value?
        nnln = n;               %yes, assign lower neighbor to it
        nnun = minn + ssizen;   %upper nearest neighbor is a stepsize up
        rnnun = rn;             %abort the nnun, nnln searches
        rnnln = rn;             %
    %n is the maximum observation
    elseif n == maxn;
        nnln = maxn - ssizen;   %go a stepsize down for nnln
        nnun = n;               %already at highest observation value
        rnnun = rn;             %abort the nnun, nnln searches
        rnnln = rn;             %
```

```
    %n is somewhere in the middle; in this case, pass all known info fwd.
    elseif n > minn && n < maxn
        nnln = n;
        nnun = n;
        rnnun = rn;
        rnnln = rn;
    end
end
%
%search column 2 for the upper/lower bounds
if isempty(rn) == 1
    ind = 0;                                %reset the search index
    while isempty(rnnun)                    %run loop while rnnup is empty
        ind = ind+1;                        %...waiting to find a match
        [rnnun] = find(S(:,2) == n+ind);    %ADD the index; search up from n
    end
    nnun = n+ind;                           %got a match! upper neighbor
%now find the lower nearest neighbor variable
    ind = 0;                                %reset the search index
    while isempty(rnnln)                    %run loop while rnnlp is empty
        ind = ind+1;                        %...waiting to find a match
        [rnnln] = find(S(:,2) == n-ind);    %SUB the index; search dn from n
    end
    nnln = n-ind;                           %got a match! lower neighbor
end
%
%check for being on the diagonal n = p
if nnln == nnlp && nnun == nnup
    nnln = nnun;            %lower takes same obs. value as upper
    [rnnln] = find(S(:,2) == nnln);
end
%truncate data one more time; subset of S meaningful to the analysis
%in between rows 2 and 3 is the solution for (getp,getn)
Y = [S(rnnln(1),:);
     S(rnnun(1),:);
     S(rnnln(2),:);
     S(rnnun(2),:)];
%*************************************************************************
%interpolation: find getn y-coordinate (mean eigenvalues) given a single
%x-coordinate.  This is a an inverse use of the interp1 function, as it
%wants a unique x value for each y.  Because the x-value is fixed along the
%curve (essentially the upper nearest neighbor and lower nearest neighbors
%are defining the curve), the interp1 routine used here has to cycle
%through a pair of points defined at the upper and lower nn observations.
%format is (ev=mean eigenvalue in all cases):
%ev we want to find = interp1([fixed lower nn obs; fixed upper nn obs],...
%                             [lower ev @ this x; upper ev @ this x],...
%                             observation we are trying to find)
%this routine handles the 'bias' factor; that is, how close getn is to
%either the upper or lower nearest neighbors impacts the
ncol = length(Y(1,3:end));  %number of y-data columns (contains mean ev's)
                            %DO NOT USE ncol FOR PLOTTING--IT IS OFF BY 2
evu = zeros(1,ncol);        %preallocate space
evl = zeros(1,ncol);        %preallocate space
getev = zeros(1,ncol);      %preallocate space
%*************************************************************************
%the logic in the following if/elseif lines evaluates four conditions for
```

```
%(p,n): The combinations that p and n are direct references in tablex.
%*************************************************************************
%(p,n) not in lookup tablex; interpolate a two-part solution.
%Part 1: Upper and lower curves.  Interpolate nnln, nnun variables for n
%Part 2: Middle curve (the answer!).  Using the curves from Part 1 as
%        input, interpolate the variable curve in between nnup and nnlp.
%OR statement is (p,n) was on diagonal and corrections made (~line 171)
if (isempty(rn) == 1 && isempty(rp) == 1) && ~(nnln == nnun)
    for i = 1:ncol;          %loop through the data; interpolate mev values
                             %for ev([u]pper) and ev([l]ower) curves
        evu(i) = interp1([nnln;nnun],[Y(1,2+i);Y(2,2+i)],n);  %upper ev's
        evl(i) = interp1([nnln;nnun],[Y(3,2+i);Y(4,2+i)],n);  %lower ev's
    end
    %getev is the solution for the curve describing (p,n)
    for i = 1:ncol
        getev(i) = interp1([nnup;nnlp],[evu(i);evl(i)],p);
    end
%p is in tablex; n is not.  Get (p,nnun) and (p,nnln) to interp a soln
elseif isempty(rn) == 1 && isempty(rp) == 0;
    for i = 1:ncol;          %loop through the data; interpolate ev's
        getev(i) = interp1([nnln;nnun],[Y(1,2+i);Y(2,2+i)],n);
    end
    evu = getev;             %already had the variable, only needed the
    evl = getev;             %interpolation on the observations
%p is not in tablex, n is.  Get (nnlp,n) and (nnup,n) to interp a soln
%OR statement is (p,n) was on diagonal and corrections made (~line 171)
elseif (isempty(rn) == 0 && isempty(rp) == 1) || nnln == nnun &&...
        ~(nnlp == nnup)
    evu = Y(1,3:end);
    evl = Y(3,3:end);
    for i = 1:ncol;
        getev(i) = interp1([nnup;nnlp],[evu(i);evl(i)],p);
    end
%(p,n) are both in the table.  Direct referenced value--no interpolation
else isempty(rn) == 0 && isempty(rp) == 0;
    evu = Y(1,3:end);
    evl = evu;
    getev = evu;
end
%
curve1 = Y(1,3:end);         %nnun mev's for nnup
curve2 = evu;                %interpolated mev's for getn on nnun
curve3 = Y(2,3:end);         %nnun mev's for nnlp
curve4 = getev;              %interpolated mev's for getp
%curve5 = mevvec(1:eind);    %from the MCS run; function EigenMean provides
curve6 = Y(3,3:end);         %nnln mev's for nnlp
curve7 = evl;                %interpolated mev's for getn on nnln
curve8 = Y(4,3:end);         %nnun mev's for nnlp
%curve5 will not be seen in the function return matrix
curves = [curve1; curve2; curve3; curve4; curve6; curve7; curve8];
%
%end of function
```

Supporting Function: findcurves2OM.m

```
function [curves,nnup,nnlp,nnun,nnln]=findcurves2OM(p,n,minp,maxp,minn,maxn)
%Original code by Captain Andrew L. Bigley, USAF.  Written for partial
%fulfillment of a Master's of Science Degree in Operations Research, The
%Air Force Institute of Technology (AFIT), Wright-Patterson AFB, OH USA
%US Government disclaimer:
%The views expressed in this thesis are those of the author and do not
%reflect the official policy or position of the United States Air Force,
%Department of Defense, or the United States Government.
%This material is declared a work of the U.S. Government and is not subject
%to copyright protection in the United States.
%21 March 2013
%
%Linear regression second-order model, Case I:
% * Both the variable (p') and observation (n') is in the range of the    *
% * lookup table mapped values.  Also, (p') and (n') are not the minimum  *
% * nor maximum values in the table and the nearest neighbors method will *
% * achieve satisfactory results.                                          *
%
%The function returns six curves to the main executable:
%1) Upper nearest neighbor variable, lower nearest neighbor observation.
%2) Interpolated curve for p', based upon upper p & lower/upper obs.
%3) Upper nearest neighbor variable, upper nearest neighbor observation.
%4) Lower nearest neighbor variable, lower nearest neighbor observation.
%6) Interpolated curve for p', based upon lower p & lower/upper obs.
%7) Lower nearest neighbor variable, upper nearest neighbor observation.
%
global tablexbeta ssizep ssizen
%
rnnup = []; %init the data row variable; also acts as a flag to search
rnnlp = []; %init the data row variable; also acts as a flag to search
rnnun = []; %init the data row variable; also acts as a flag to search
rnnln = []; %init the data row variable; also acts as a flag to search
%*************************************************************************
%******Check to see if data is already in the lookup table and at what*****
%*extremes.  Data at or near an edge will need to be conditioned to accept*
%*something else than what the nearest neighbor search algorithm assigns***
%******Variable assignment***********************************************
[rp] = find(tablexbeta(:,1) == p);  %look for the input variable
if isempty(rp) == 0;            %found p in tablex but where is it?
    %p is the minimum variable
    if p == minp;               %is the input = min table variable value?
        nnlp = p;               %yes, assign lower neighbor to it
        nnup = minp + ssizep;   %upper nearest neighbor is a stepsize up
        rnnup = rp;             %abort the nnup, nnlp searches
        rnnlp = rp;             %
    %p is the maximum variable
    elseif p == maxp;
        nnlp = maxp - ssizep;   %go a stepsize down for nnlp
        nnup = p;               %already at highest variable value
        rnnup = rp;             %abort the nnup, nnlp searches
        rnnlp = rp;             %
    %p is somewhere in the middle; in this case, pass all known info fwd.
    elseif p > minp && p < maxp
        nnlp = p;
```

```
        nnlp = p;
        nnup = p;
        rnnup = rp;
        rnnlp = rp;
    end
end
%
if isempty(rp) == 1
    ind = 0;                                    %reset the search index
    while isempty(rnnup) && isempty(rp)         %run loop while empty
        ind = ind+1;                            %incr the array counter
        [rnnup] = find(tablexbeta(:,1) == p+ind);   %add the index
        if ind > ssizep
            fprintf('WARNING: Variable stepsize exceeded. Check tablex.\n')
            rp = -1;    %value indicates we had a problem here.
            break
        end
    end
    nnup = p+ind;                               %got the upper neighbor
    %now find the lower nearest neighbor variable
    ind = 0;                                    %reset the search index
    while isempty(rnnlp) && isempty(rp)         %run loop while empty
        ind = ind+1;                            %incr the array counter
        [rnnlp] = find(tablexbeta(:,1) == p-ind);   %subtract the index
        if ind > ssizep
            fprintf('WARNING: Variable stepsize exceeded. Check tablex.\n')
            rp = -2;    %value indicates we had a problem here.
            break
        end
    end
    nnlp = p-ind;                               %got the lower neighbor
end             %exit out of looking for the variable neighbor indices
%
%truncate the data into a subset of tablex meaningful to the analysis
S = [tablexbeta(rnnup,:); tablexbeta(rnnlp,:)];
%S is the reduced matrix to work from for observation nearest neighbors****
%***********************Done with VARIABLES*******************************
%***find the OBSERVATION nearest neighbors in the truncated data matrix S**
%find the start of zero entries in the lower bound (nnlp) and then trim
%each bound (upper and lower) to that length.  Purpose: set equal number of
%variables in the vectors of matrix S
[rn] = find(S(:,2) == n);       %look for the input observation
if isempty(rn) == 0;            %found n in S but where is it?
    %n is the minimum observation
    if n == minn;               %is the input = min(S) value?
        nnln = n;               %yes, assign lower neighbor to it
        nnun = minn + ssizen;   %upper nearest neighbor is a stepsize up
        rnnun = rn;             %abort the nnun, nnln searches
        rnnln = rn;             %
    %n is the maximum observation
    elseif n == maxn;
        nnln = maxn - ssizen;   %go a stepsize down for nnln
        nnun = n;               %already at highest observation value
        rnnun = rn;             %abort the nnun, nnln searches
        rnnln = rn;             %
    %n is somewhere in the middle; in this case, pass all known info fwd.
    elseif n > minn && n < maxn
```

```
        nnln = n;
        nnun = n;
        rnnun = rn;
        rnnln = rn;
    end
end
%
%search column 2 for the upper/lower bounds
if isempty(rn) == 1
    ind = 0;                                %reset the search index
    while isempty(rnnun)                    %run loop while rnnup is empty
        ind = ind+1;                        %...waiting to find a match
        [rnnun] = find(S(:,2) == n+ind);    %ADD the index; search up from n
    end
    nnun = n+ind;                           %got a match! upper neighbor
%now find the lower nearest neighbor variable
    ind = 0;                                %reset the search index
    while isempty(rnnln)                    %run loop while rnnlp is empty
        ind = ind+1;                        %...waiting to find a match
        [rnnln] = find(S(:,2) == n-ind);    %SUB the index; search dn from n
    end
    nnln = n-ind;                           %got a match! lower neighbor
end
%
%check for being on the diagonal n = p
if nnln == nnlp && nnun == nnup
    nnln = nnun;            %lower takes same obs. value as upper
    [rnnln] = find(S(:,2) == nnln);
end
%truncate data one more time; subset of S meaningful to the analysis
%in between rows 2 and 3 is the solution for (getp,getn)
Y = [S(rnnln(1),:);
     S(rnnun(1),:);
     S(rnnln(2),:);
     S(rnnun(2),:)];
%*************************************************************************
%interpolation: find getn y-coordinate (coefficients) given a single
%x-coordinate.  This is a an inverse use of the interp1 function, as it
%wants a unique x value for each y.  Because the x-value is fixed along the
%curve (essentially the upper nearest neighbor and lower nearest neighbors
%are defining the curve), the interp1 routine used here has to cycle
%through a pair of points defined at the upper and lower nn observations.
%format is (ev=mean eigenvalue in all cases):
%ev we want to find = interp1([fixed lower nn obs; fixed upper nn obs],...
%                             [lower ev @ this x; upper ev @ this x],...
%                             observation we are trying to find)
%this routine handles the 'bias' factor; that is, how close getn is to
%either the upper or lower nearest neighbors impacts the
%ncol = length(Y(1,3:end));  %number of y-data columns (contains coeffs)
                            %DO NOT USE ncol FOR PLOTTING--IT IS OFF BY 2
ncol = 3;                   %already know how many columns have beta val.
cu = zeros(1,ncol);         %preallocate space
cl = zeros(1,ncol);         %preallocate space
getbeta = zeros(1,ncol);    %preallocate space
%*************************************************************************
%the logic in the following if/elseif lines evaluates four conditions for
%(p,n): The combinations that p and n are direct references in tablex.
```

```
%*************************************************************************
%(p,n) not in lookup tablex; interpolate a two-part solution.
%Part 1: Upper and lower curves.  Interpolate nnln, nnun variables for n
%Part 2: Middle curve (the answer!).  Using the curves from Part 1 as
%        input, interpolate the variable curve in between nnup and nnlp.
%OR statement is (p,n) was on diagonal and corrections made (~line 171)
if (isempty(rn) == 1 && isempty(rp) == 1) && ~(nnln == nnun)
    for i = 1:ncol;          %loop through the data; interpolate beta values
                             %for coeff([u]pper) and coeff([l]ower) curves
        cu(i) = interp1([nnln;nnun],[Y(1,2+i);Y(2,2+i)],n);  %upper
        cl(i) = interp1([nnln;nnun],[Y(3,2+i);Y(4,2+i)],n);  %lower
    end
    %getev is the solution for the curve describing (p,n)
    for i = 1:ncol
        getbeta(i) = interp1([nnup;nnlp],[cu(i);cl(i)],p);
    end
%p is in tablex; n is not.  Get (p,nnun) and (p,nnln) to interp a soln
elseif isempty(rn) == 1 && isempty(rp) == 0;
    for i = 1:ncol;          %loop through the data; interpolate ev's
        getbeta(i) = interp1([nnln;nnun],[Y(1,2+i);Y(2,2+i)],n);
    end
    cu = getbeta;             %already had the variable, only needed the
    cl = getbeta;             %interpolation on the observations
%p is not in tablex, n is.  Get (nnlp,n) and (nnup,n) to interp a soln
%OR statement is (p,n) was on diagonal and corrections made (~line 171)
elseif (isempty(rn) == 0 && isempty(rp) == 1) || nnln == nnun &&...
        ~(nnlp == nnup)
    cu = Y(1,3:5);
    cl = Y(3,3:5);
    for i = 1:ncol;
        getbeta(i) = interp1([nnup;nnlp],[cu(i);cl(i)],p);
    end
%(p,n) are both in the table.  Direct referenced value--no interpolation
else isempty(rn) == 0 && isempty(rp) == 0;
    cu = Y(1,3:5);
    cl = cu;
    getbeta = cu;
end
%
%return the curve describing the lines, not just the coefficients of the
%model
xfine = 1:p;                    %0.1 controls the fidelity in the curve
                                %to decrease, try 0.2 to 0.5
curve1 = polyval(Y(1,3:5),xfine);   %nnun betas for nnup
curve2 = polyval(cu,xfine);         %interpolated betas for getn on nnun
curve3 = polyval(Y(2,3:5),xfine);   %nnun betas for nnlp
curve4 = polyval(getbeta,xfine);    %interpolated betas for getp
%curve 5 not used in the 2OM evaluation
curve6 = polyval(Y(3,3:5),xfine);   %nnln betas for nnlp
curve7 = polyval(cl,xfine);         %interpolated betas for getn on nnln
curve8 = polyval(Y(4,3:5),xfine);   %nnun betas for nnlp
curves = [curve1; curve2; curve3; curve4; curve6; curve7; curve8];
%
%end of function
```

Horn's Curve Estimation Through Multi-Dimensional Interpolation

Capt Andrew L. Bigley
Advisor: Dr. Kenneth W. Bauer
Reader: Lt Col Mark A. Friend
Department of Operational Sciences (ENS)
Air Force Institute of Technology

Introduction

- Principal Components (PCs) Analysis (PCA) is a data reduction technique used to summarize multivariate information through hidden structure (components)
- Each component contains a portion of total variance; those considered significant are 'principal' and are extracted for further analysis
- Determining how many PCs to extract is not trivial
  - Too many leads to meaningless linear combinations
  - Too few discards summary information

Research Objectives

- Develop a graphical software tool to determine accurately the number PCs to extract
- Remove subjectivity on behalf of the analyst; offer flexible, parsimonious solutions-of-value
- Incorporate supplementary visual and tabular information

AIR FORCE INSTITUTE OF TECHNOLOGY

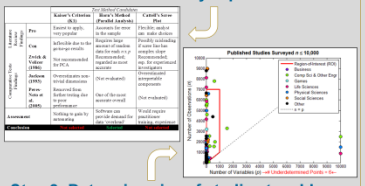
Methodology

Criteria for a candidate stopping rule:

- Visual, Accurate dimensionality estimator, Leads to objective assessments

Step 1: Survey the literature for candidates

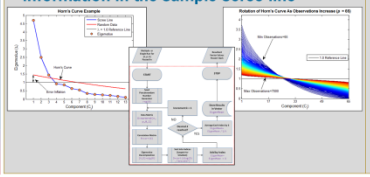
- Three techniques located; Horn's test selected due to accuracy & positive eval.



Step 2: Determine size of studies to address

- Number and sizes of published research indicate region-of-interest (ROI) is within $5 \le p \le 1000$ and $5 \le n \le 7000$ (captures 80.3% of the 178 studies surveyed)

Step 3: Develop Horn's test theory into MATLAB algorithms. Horn's methodology separates *noise* from *signal* by considering inherent random error from useful information in the sample scree line

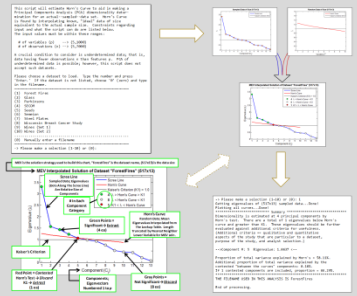


- Horn's methodology requires two distinct elements (sampled & random data)
  - The final solution is a synthesis of *both* brought together
- Two solution strategies, similar methods
  - Mean eigenvalues
  - Linear regression second-order model
- Monte Carlo simulation of random data

$$x_{ij} \sim NID(0, \mathbf{I}_p) \text{ where } \begin{cases} i \in \{1,2,\ldots,n-1,n\} \\ j \in \{1,2,\ldots,p-1,p\} \end{cases} \rightarrow \mathbf{X}_{n \times p} = \begin{bmatrix} x_{11} & \cdots & x_{1p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{bmatrix}$$

- Eigendecomposition of corr($\mathbf{X}_{n \times p}$)
- Lookup table of sparse, preprocessed data (196 hrs CPU time for 26,650 x 1002 matrix, 80 MB; regression compacts to < 1 MB)
- Lookup tables nearest neighbor search
- Two-part piecewise linear interpolation

Step 4: Bring both elements together & display graphical and tabulated summary



Contributions

- Successfully automated the PC extraction stopping rule Horn's test (Parallel Analysis)
- Dimensionality estimate presented clearly in graphical and tabulated formats
  - Visual analysis includes familiar features of other, common stopping rules
  - Output for percent variance summarized by the PCs plus any 'contested' components
- Analyst allowed flexibility to combine 'art with science' in reaching an informed assessment
  - Balances removal of unwarranted subjectivity with too rigid of an analytical result

Future Research

- Confirmatory analysis of algorithm accuracy using datasets of designed structure
- Possibility of artificial neural network to learn the ROI
- Discovery of 'pathological' type datasets and how the algorithms should handle them

Bibliography

Bauer, K. W. (2012, March). Course Notes. *OPER 685, Applied Multivariate Analysis I.* Wright-Patterson Air Force Base, OH, USA: The Air Force Institute of Technology.

Cattell, R. B. (1966). The Scree Test for The Number of Factors. *Multivariate Behavioral Research*, 245-276.

Charytanowicz, M., & Niewczas, J. (2012). Seeds Dataset (primary co-acknowledged). Lublin, Poland: Institute of Mathematics and Computer Science, The John Paul II Catholic University of Lublin. Retrieved January 14, 2013, from http://archive.ics.uci.edu/ml/datasets/seeds

Cortez, P., & Morais, A. (2007). A Data Mining Approach to Predict Forest Fires Using Meteorlogical Data. In J. Neves, M. F. Santos, & J. Machado (Ed.), *New Trends in Artificial Intelligence* (pp. 512-523). Guimares: APPIA (ISBN-13 978-989-95618-0-9). Retrieved 2013 14, January

Cortez, P., Cerdeira, A., Almeida, F., Matos, T., & Reis, J. (2009). Modeling Wine Preferences by Data Mining from Physicochemical Properties. *Decision Support Systems, 47*(4), 547-553. Retrieved January 14, 2013, from http://archive.ics.uci.edu/ml/datasets/Wine+Quality

Dillon, W. R., & Goldstein, M. (1984). *Multivariate Analysis.* New York: John Wiley & Sons.

Ehrenberg, A., & Goodhardt, G. (1976). *Factor Analysis: Limitations and Alternatives.* Cambridge, MA: Marketing Science Institute.

Frank, A., & Asuncion, A. (2010). *University of California Machine Learning Repository.* Retrieved from School of Information and Computer Science.

Franklin, S. B., Gibson, D. J., Robertson, P. A., Pohlmann, J. T., & Fralish, J. S. (1995). Parallel Analysis: A Method for Determining Significant Principle Components. *Journal of Vegetation Science*, 99-106.

Horn, J. L. (1965). A Rationale and Test for The Number of Factors in Factor Analysis. *Psychometrika*, 179-185.

Horn, J. L., & Engstrom, R. (1979). Cattell's Scree Test in Relation to Bartlett's Chi-Square Test and Other Observations on The Number of Factors Problem. *Multivariate Behavioral Research*, 283-300.

Jackson, D. A. (1993). Stopping Rules in Principal Components Analysis: A Comparison of Heuristical and Statistical Approaches. *Ecology*, 2204-2214.

Kaiser, H. F. (1960). The Application of Electronic Computers to Factor Analysis. *Educational and Psychological Measurement*, 141-151.

Kaiser, H. F. (1986, October 6). The Application of Electronic Computers to Factor Analysis. *Citation Classics*, p. 18.

Kulczycki, P., Kowalski, P., Lukasik, S., & Zak, S. (2012). Seeds Dataset (secondary co-acknowledgment). Warsaw, Poland: Systems Research Insitute, Polish Academy of Sciences.

Little, M. A., McSharry, P. E., Roberts, S. J., Costello, D. A., & Moroz, I. M. (2007). Exploiting Nonlinear Recurrent and Fractal Scaling Properties for Voice Disorder Detection. *BioMedical Engineering OnLine, 6*(23). Retrieved January 14, 2013, from http://archive.ics.uci.edu/ml/datasets/Parkinsons

Matsumoto, M. (2011, June 20). *Mersenne Twister Home Page*. Retrieved January 8, 2013, from Home Page of Makoto Matsumoto: http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html

Montgomery, D. C., Peck, E. A., & Vining, G. G. (2006). *Introduction to Linear Regression Analysis (Fourth Edition).* Hoboken: John Wiley & Sons.

Peres-Neto, P. R., Jackson, D. A., & Somers, K. A. (2005). How Many Principal Components? Stopping Rules for Determining The Number of Non-Trivial Axes Revisited. *Computational Statistics and Data Analysis*, 974-997.

Quateroni, A., & Saleri, F. (2003). *Scientific Computing with MATLAB.* Berlin: Springer-Verlag.

Recktenwald, G. (2000). *Numerical Methods with MATLAB: Implementation and Application.* Upper Saddle River: Prentice Hall.

Ross, S. M. (2007). *Introduction to Probability Models (Ninth Edition).* Burlington: Academic Press.

Sawilowky, S. S. (2003). You Think You've Got Trivials? *Journal of Modern Applied Statistical Methods*, 218-225.

Semeion Research Center for the Science of Communication. (2008, November 11). Semeion Handwritten Digit Data Set. Rome, Italy. Retrieved January 12, 2012, from http://archive.ics.uci.edu/ml/datasets/Semeion+Handwritten+Digit

Semeion Research Center for the Science of Communication. (2010, October 26). Steel Plates Faults. Rome, Italy. Retrieved January 13, 2013, from http://archive.ics.uci.edu/ml/datasets/Steel+Plates+Faults

The MathWorks. (2012, September 11). *MATLAB Online Documentation Center*. Retrieved January 23, 2013, from The MathWorks Website: http://www.mathworks.com/help/matlab/random-number-generation.html

University of California-Irvine. (2007). *Machine Learning Repository*. Retrieved January 8, 2013, from Center for Machine Learning and Intelligent Systems.

Velicer, W. F. (1976). Determining The Number of Components from The Matrix of Partial Correlations. *Psychometrika*, 321-327.

Wolberg, W. H. (1992, July 15). Breast Cancer Wisconsin (Original) Data Set. Madison, Wisconsin, USA. Retrieved January 14, 2013, from http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29

Wolberg, W. H., & Mangasarian, O. L. (1990). Multisurface Method of Pattern Separation for Medical Diagnosis Applied to Breast Cytology. *National Academy of Sciences, 87*, pp. 9193-9196.

Zwick, W. R., & Velicer, W. F. (1986). Comparison of Five Rules for Determining The Number of Components to Retain. *Psychological Bulletin*, 432-442.

REPORT DOCUMENTATION PAGE		*Form Approved* *OMB No. 0704–0188*
The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704–0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202–4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.		
1. REPORT DATE (DD–MM–YYYY) 21-03-2013	2. REPORT TYPE Master's Thesis	3. DATES COVERED *(From — To)* June 2012 – March 2013
4. TITLE AND SUBTITLE Horn's Curve Estimation Through Multi-Dimensional Interpolation		5a. CONTRACT NUMBER 5b. GRANT NUMBER 5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Bigley, Andrew L., Captain, USAF		5d. PROJECT NUMBER 5e. TASK NUMBER 5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way WPAFB OH 45433-7765		8. PERFORMING ORGANIZATION REPORT NUMBER AFIT-ENS-13-M-01
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Intentionally left blank		10. SPONSOR/MONITOR'S ACRONYM(S) n/a 11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION / AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED		
13. SUPPLEMENTARY NOTES This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.		
14. ABSTRACT: A well-known multivariate data reduction method is principal components analysis (PCA). PCA transforms the variables under study into a set of components that are used to summarize the variation among the variables. The benefit is the dimension of the data may be reduced by the descriptive power of the components, permitting tractable analysis on large and messy datasets. Integral to successful PCA is determining when to stop extracting components – the matter is not a trivial one. A component extraction stopping rule that consistently produces reliable estimates of principal components is Horn's test. The drawback of the test is it requires a large amount of random data to evaluate the hidden component structure. Leveraging the flexibility and power of the MATLAB software package, a lookup table interpolates nearest neighbor searches of pre-processed mean eigenvalue data to provide real-time results for datasets up to 1,000 variables on 7,000 samples. The methodology is extended to a linear regression second-order model producing Horn's curve, significantly reducing the required size of the lookup table with no loss of resolution into the dimensionality estimate.		
15. SUBJECT TERMS Principal components analysis, Horn's test, dimensionality, component extraction stopping rules, interpolation		

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON Dr. Kenneth W. Bauer/ENS
a. REPORT U	b. ABSTRACT U	c. THIS PAGE U	UU	157	19b. TELEPHONE NUMBER (Include Area Code) (937) 255-3636, ext 4328

Standard Form 298 (Rev. 8–98)
Prescribed by ANSI Std. Z39.18